get –u gorm.io/gorm)

(go get –u gorm.io/driver/mysql)
* goini:[ini.unknwon.io/](http://cxyroad.com/)

(go get –u gopkg.in/ini.v1)
* jwt:[pkg.go.dev/github.com/…](http://cxyroad.com/
”https://pkg.go.dev/github.com/golang-jwt/jwt/v5”)

(go get –u [github.com/golang-jwt/…](http://cxyroad.com/
”https://github.com/golang-jwt/jwt”))
* cors：[pkg.go.dev/github.com/…](http://cxyroad.com/
”https://pkg.go.dev/github.com/rs/cors”)

(go get –u [github.com/rs/cors](http://cxyroad.com/
”https://github.com/rs/cors”))
* rotatelogs：[pkg.go.dev/github.com/…](http://cxyroad.com/
”https://pkg.go.dev/github.com/lestrrat-go/file-rotatelogs”)

(go get –u github.com/lestrrat-go/file-rotatelogs)
* lfshook：[pkg.go.dev/github.com/…](http://cxyroad.com/
”https://pkg.go.dev/github.com/rifflock/lfshook”)

(go get –u [github.com/rifflock/lf…] de,
”data”: data,
”message”: errmsg.GetErrMsg(code),
})
}

```

然后进入到AddUser的结构体，这里会和数据库做个绑定，然后判断用户名存不存在，符不符合要求，接着和model文件夹的User.go文件的CreateUser进行数据交换

```
func CreateUser(data *User) int {

```
//加密密码
salt := make([]byte, 8)
if _, err := rand.Read(salt); err != nil {
log.Fatal(err)
}
data.Salt = base64.StdEncoding.EncodeToString(salt) // 存储盐值
data.Password = ScryptPw(data.Password, data.Salt)  // 使用盐值加密密码
err := db.Create(&data).Error
if err != nil {
log.Printf("Error CreateUser: %v", err)
return errmsg.ERROR
}
return errmsg.SUCCESS
}
```

这里进入了CreateUser，先加密了密码，然后存进了mysql数据库

总结
--

本人没用过其他语言写项目，不过用go语言写后端，代码逻辑和思路很清晰。

这是个人博客项目，go语言后端充当了前端和mysql数据库之间的桥梁，Go语言后端负责处理来自前端的HTTP请求，执行逻辑处理，与数据库进行交互（执行SQL命令进行增删改查），然后将结果返回给前端。