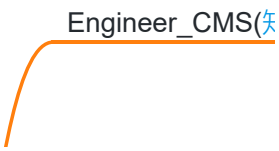
目　录

概述
第一步 文件夹合并
第二部 engineercms自身文件的修改
第三步 mindoc文件的修改
第四步 升级
5 20260916记录
5 jstree和ajax请求-添加上一页下一页
6 为何只支持一张图片复制粘贴上传
7 批量上传图片
8 添加froala富文本编辑器
9 为开源做贡献
10 添加目录访问密码
11 集成onlyoffice

概述

对mindoc向往已久

1. 这种可以在网络上广泛传播的电子书，特点很明显，就是左侧是目录，右侧是图文并茂。类似word的导航栏，但是，word在网页上没法这样排布。
2. 另外就是支持全文检索。
3. 这次终于有时间将它整合到我的项目中来，也契合我的初衷：即，创作知识。
4. 整合mindoc的本质目的，就是方便大家在engineercms里直接写书（创作知识），这种形式是其他方式比较难代替的，比如博客形式，它是一篇篇的文章，不方便组合成一部分量较重的书籍；还有就是方便传播，即不同地方方便导入导出。
5. 经过我的代码贡献，mindoc现在支持froala富文本编辑器，该编辑器支持word图文转存，也支持web图文转存，相比以前一张张图片复制黏贴，大大节约了时间；支持markdown等编辑器批量插入图片，符合“新质生产力”要求。



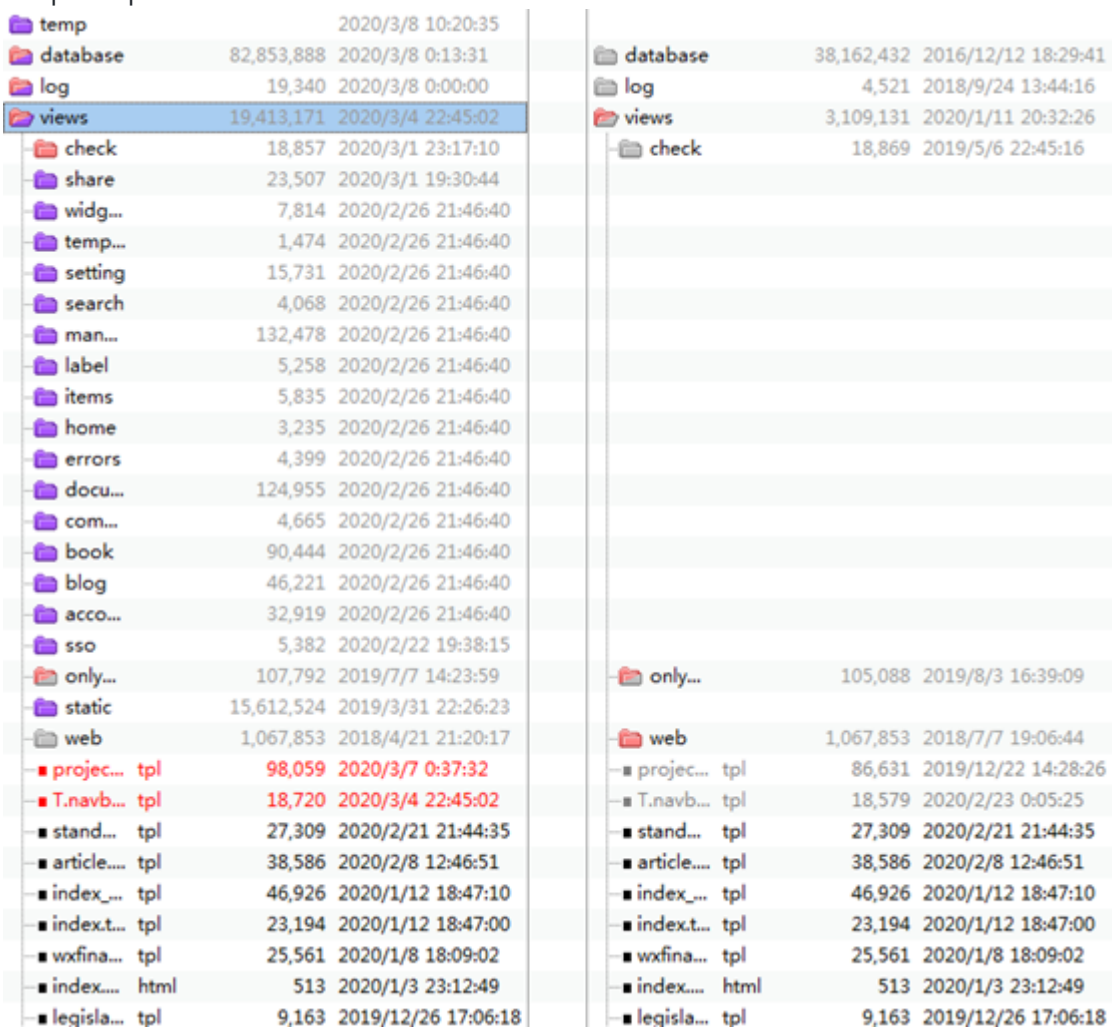
第一步 文件夹合并

直接下载mindoc

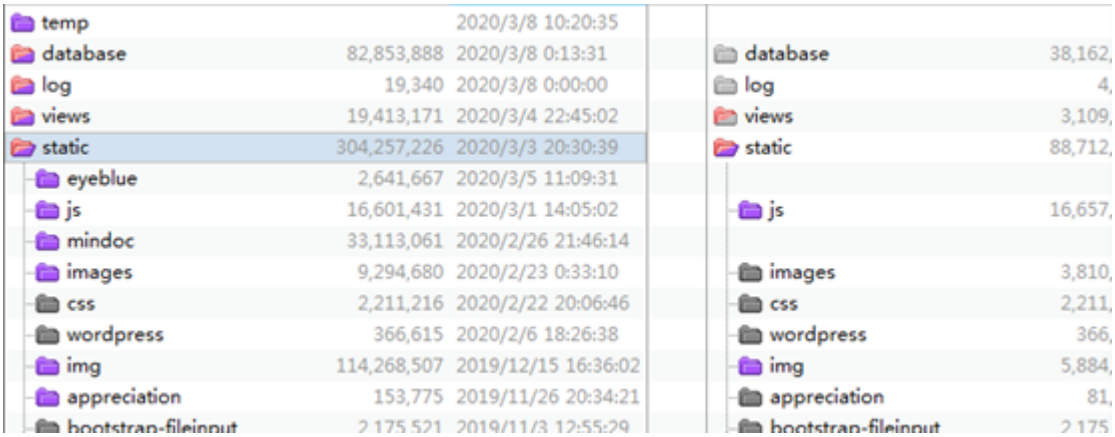
因为都是beego框架项目，可以简单地将文件和文件夹与我项目的文件夹合并。

1.1 view目录

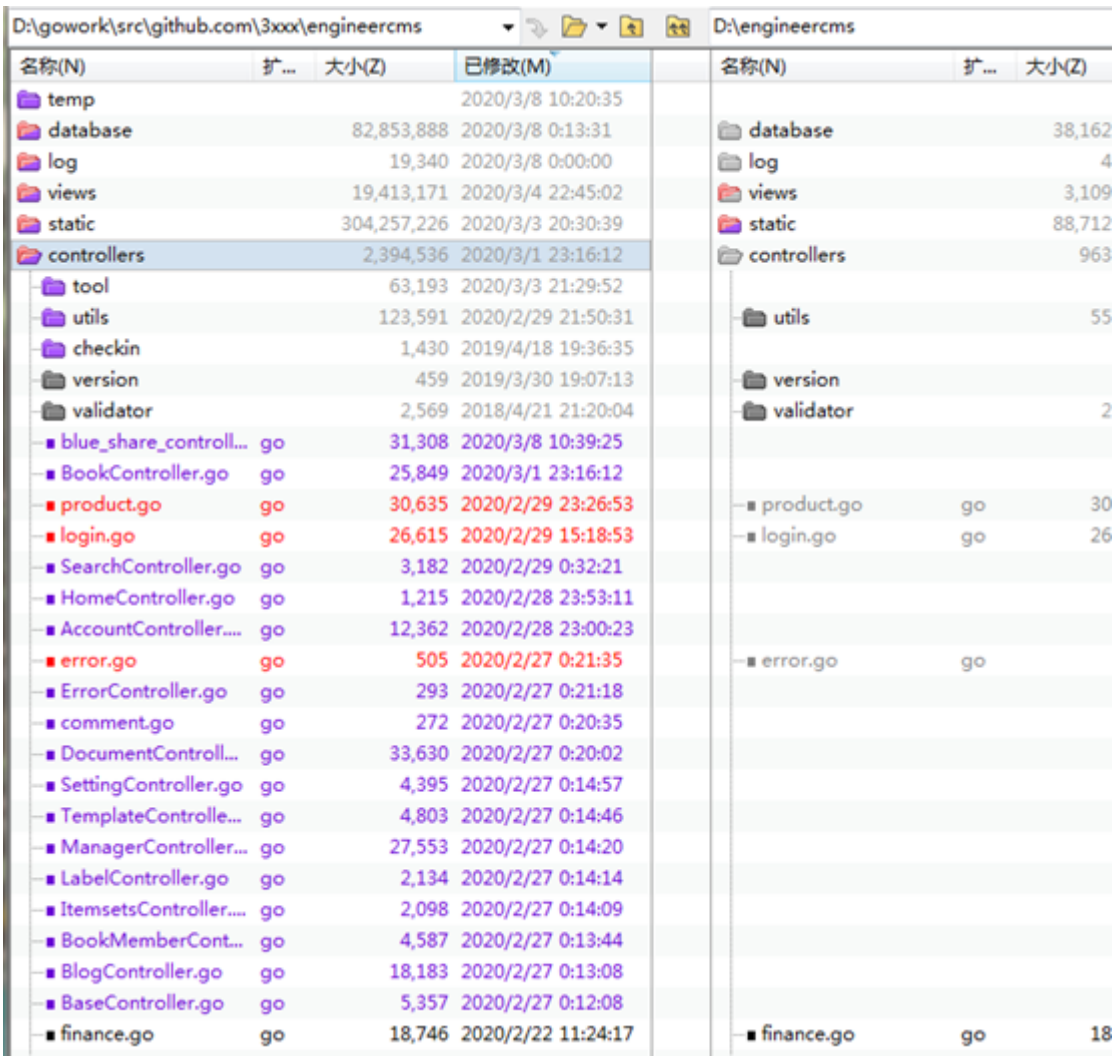
template.tpl



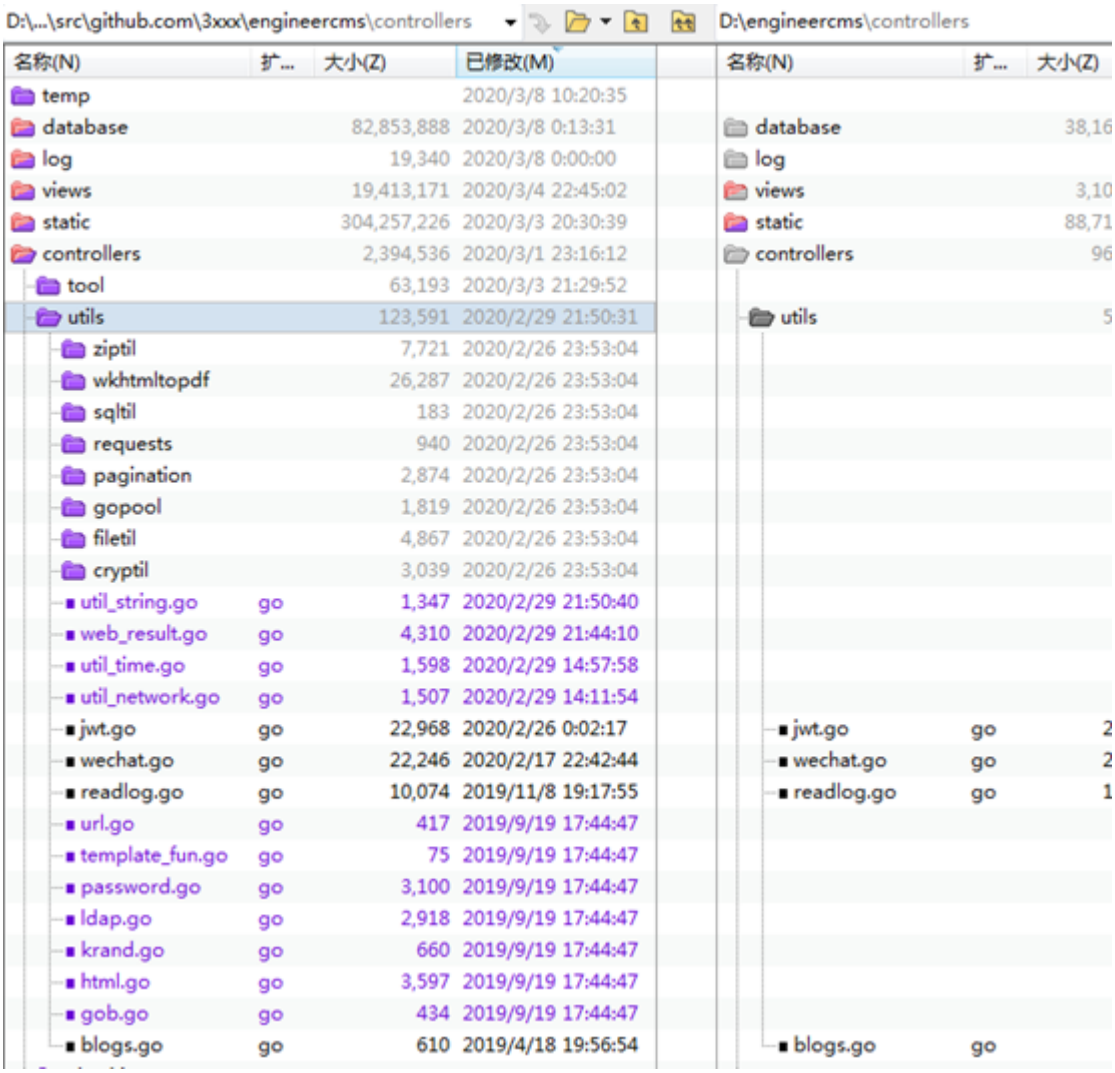
1.2 static



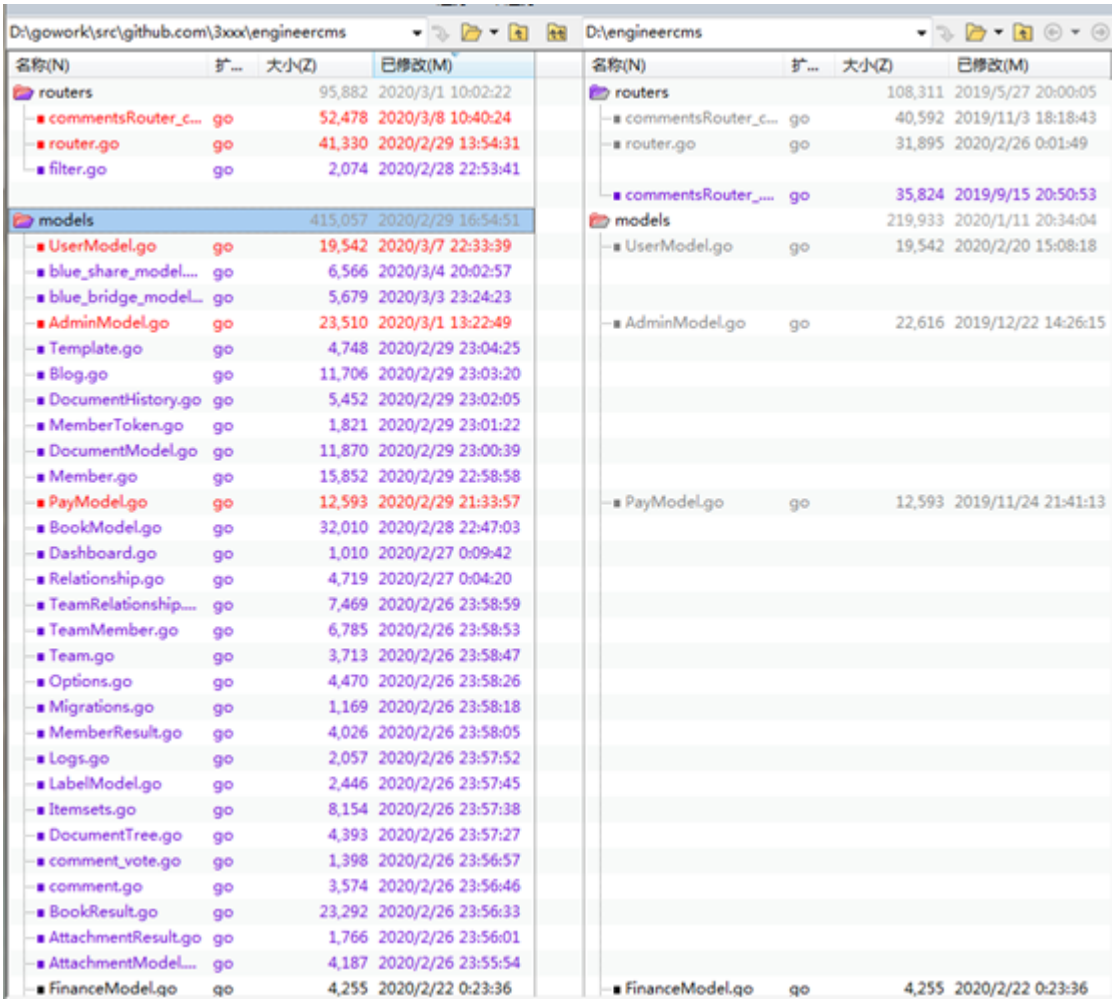
1.3 controllers



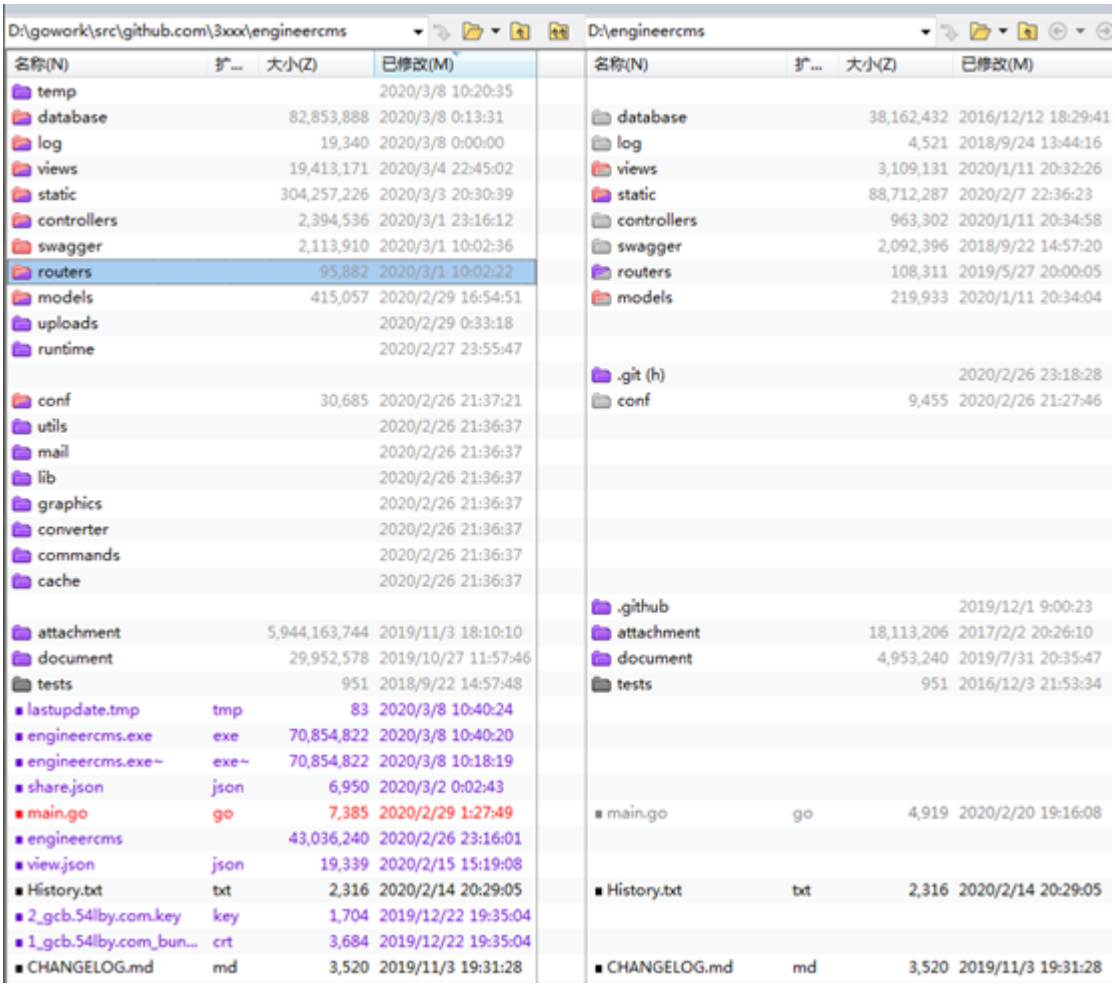
1.3.1 utils



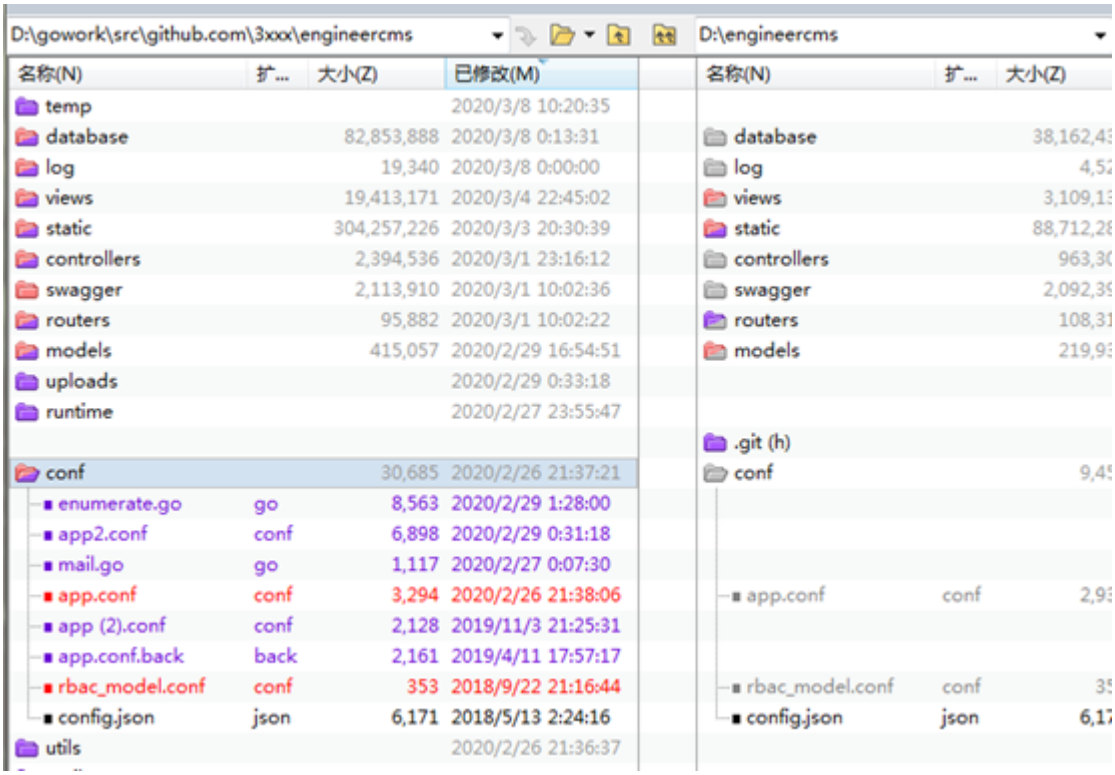
1.4 models和route


D:\gowork\src\github.com\3xxx\engineercms
D:\engineercms
名称(N) 扩... 大小(Z) 已修改(M)
routers 95,882 2020/3/1 10:02:22
commentsRouter_c... go 52,478 2020/3/8 10:40:24
router.go go 41,330 2020/2/29 13:54:31
filter.go go 2,074 2020/2/28 22:53:41
models 415,057 2020/2/29 16:54:51
UserModel.go go 19,542 2020/3/7 22:33:39
blue_share_model.... go 6,566 2020/3/4 20:02:57
blue_bridge_model... go 5,679 2020/3/3 23:24:23
AdminModel.go go 23,510 2020/3/1 13:22:49
Template.go go 4,748 2020/2/29 23:04:25
Blog.go go 11,706 2020/2/29 23:03:20
DocumentHistory.go go 5,452 2020/2/29 23:02:05
MemberToken.go go 1,821 2020/2/29 23:01:22
DocumentModel.go go 11,870 2020/2/29 23:00:39
Member.go go 15,852 2020/2/29 22:58:58
PayModel.go go 12,593 2020/2/29 21:33:57
BookModel.go go 32,010 2020/2/28 22:47:03
Dashboard.go go 1,010 2020/2/27 0:09:42
Relationship.go go 4,719 2020/2/27 0:04:20
TeamRelationship.... go 7,469 2020/2/26 23:58:59
TeamMember.go go 6,785 2020/2/26 23:58:53
Team.go go 3,713 2020/2/26 23:58:47
Options.go go 4,470 2020/2/26 23:58:26
Migrations.go go 1,169 2020/2/26 23:58:18
MemberResult.go go 4,026 2020/2/26 23:58:05
Logs.go go 2,057 2020/2/26 23:57:52
LabelModel.go go 2,446 2020/2/26 23:57:45
Itemsets.go go 8,154 2020/2/26 23:57:38
DocumentTree.go go 4,393 2020/2/26 23:57:27
comment_vote.go go 1,398 2020/2/26 23:56:57
comment.go go 3,574 2020/2/26 23:56:46
BookResult.go go 23,292 2020/2/26 23:56:33
AttachmentResult.go go 1,766 2020/2/26 23:56:01
AttachmentModel.... go 4,187 2020/2/26 23:55:54
FinanceModel.go go 4,255 2020/2/22 0:23:36
routers 108,311 2019/5/27 20:00:05
commentsRouter_c... go 40,592 2019/11/3 18:18:43
router.go go 31,895 2020/2/26 0:01:49
commentsRouter_.... go 35,824 2019/9/15 20:50:53
models 219,933 2020/1/11 20:34:04
UserModel.go go 19,542 2020/2/20 15:08:18
AdminModel.go go 22,616 2019/12/22 14:26:15
PayModel.go go 12,593 2019/11/24 21:41:13
FinanceModel.go go 4,255 2020/2/22 0:23:36
D:\gowork\src\github.com\3xxx\engineercms\models
D:\engineercms\models
LabelModel.go go 2,446 2020/2/26 23:57:45
Itemsets.go go 8,154 2020/2/26 23:57:38
DocumentTree.go go 4,393 2020/2/26 23:57:27
comment_vote.go go 1,398 2020/2/26 23:56:57
comment.go go 3,574 2020/2/26 23:56:46
BookResult.go go 23,292 2020/2/26 23:56:33
AttachmentResult.go go 1,766 2020/2/26 23:56:01
AttachmentModel.go go 4,187 2020/2/26 23:55:54
FinanceModel.go go 4,255 2020/2/22 0:23:36
CheckinModel.go go 20,288 2020/2/16 21:42:05
ProdModel.go go 11,396 2020/1/29 16:08:14
DiaryModel.go go 4,111 2019/12/15 18:07:27
ToolsModel.go go 3,799 2019/11/16 23:43:38
SearchModel.go go 7,729 2019/11/12 22:23:56
StandardModel.go go 10,767 2019/10/27 11:31:19
ArticModel.go go 3,129 2019/9/22 10:13:16
Errors.go go 2,127 2019/9/19 17:44:47
DocumentSearchResult.go go 10,529 2019/9/19 17:44:47
ConvertBookResult.go go 134 2019/9/19 17:44:47
comment_result.go go 1,027 2019/9/19 17:44:47
BlogResult.go go 43 2019/9/19 17:44:47
Base.go go 38 2019/9/19 17:44:47
OnlyofficeModel.go go 13,795 2019/8/3 12:58:46

1.5根目录


D:\gowork\src\github.com\3xxx\engineercms
名称(N) 扩... 大小(Z) 已修改(M)
temp 2020/3/8 10:20:35
database 82,853,888 2020/3/8 0:13:31
log 19,340 2020/3/8 0:00:00
views 19,413,171 2020/3/4 22:45:02
static 304,257,226 2020/3/3 20:30:39
controllers 2,394,536 2020/3/1 23:16:12
swagger 2,113,910 2020/3/1 10:02:36
routers 95,882 2020/3/1 10:02:22
models 415,057 2020/2/29 16:54:51
uploads 2020/2/29 0:33:18
runtime 2020/2/27 23:55:47
conf 30,685 2020/2/26 21:37:21
utils 2020/2/26 21:36:37
mail 2020/2/26 21:36:37
lib 2020/2/26 21:36:37
graphics 2020/2/26 21:36:37
converter 2020/2/26 21:36:37
commands 2020/2/26 21:36:37
cache 2020/2/26 21:36:37
attachment 5,944,163,744 2019/11/3 18:10:10
document 29,952,578 2019/10/27 11:57:46
tests 951 2018/9/22 14:57:48
lastupdate.tmp tmp 83 2020/3/8 10:40:24
engineercms.exe exe 70,854,822 2020/3/8 10:40:20
engineercms.exe~ exe~ 70,854,822 2020/3/8 10:18:19
share.json json 6,950 2020/3/2 0:02:43
main.go go 7,385 2020/2/29 1:27:49
engineercms 43,036,240 2020/2/26 23:16:01
view.json json 19,339 2020/2/15 15:19:08
History.txt txt 2,316 2020/2/14 20:29:05
2_gcb.54lby.com.key key 1,704 2019/12/22 19:35:04
1_gcb.54lby.com_bun... crt 3,684 2019/12/22 19:35:04
CHANGELOG.md md 3,520 2019/11/3 19:31:28
D:\engineercms
名称(N) 扩... 大小(Z) 已修改(M)
database 38,162,432 2016/12/12 18:29:41
log 4,521 2018/9/24 13:44:16
views 3,109,131 2020/1/11 20:32:26
static 88,712,287 2020/2/7 22:36:23
controllers 963,302 2020/1/11 20:34:58
swagger 2,092,396 2018/9/22 14:57:20
routers 108,311 2019/5/27 20:00:05
models 219,933 2020/1/11 20:34:04
.git (h) 2020/2/26 23:18:28
conf 9,455 2020/2/26 21:27:46
.github 2019/12/1 9:00:23
attachment 18,113,206 2017/2/2 20:26:10
document 4,953,240 2019/7/31 20:35:47
tests 951 2016/12/3 21:53:34
main.go go 4,919 2020/2/20 19:16:08
History.txt txt 2,316 2020/2/14 20:29:05
CHANGELOG.md md 3,520 2019/11/3 19:31:28

1.6 conf


D:\gowork\src\github.com\3xxx\engineercms
名称(N) 扩... 大小(Z) 已修改(M)
temp 2020/3/8 10:20:35
database 82,853,888 2020/3/8 0:13:31
log 19,340 2020/3/8 0:00:00
views 19,413,171 2020/3/4 22:45:02
static 304,257,226 2020/3/3 20:30:39
controllers 2,394,536 2020/3/1 23:16:12
swagger 2,113,910 2020/3/1 10:02:36
routers 95,882 2020/3/1 10:02:22
models 415,057 2020/2/29 16:54:51
uploads 2020/2/29 0:33:18
runtime 2020/2/27 23:55:47
conf 30,685 2020/2/26 21:37:21
enumerate.go go 8,563 2020/2/29 1:28:00
app2.conf conf 6,898 2020/2/29 0:31:18
mail.go go 1,117 2020/2/27 0:07:30
app.conf conf 3,294 2020/2/26 21:38:06
app (2).conf conf 2,128 2019/11/3 21:25:31
app.conf.back back 2,161 2019/4/11 17:57:17
rbac_model.conf conf 353 2018/9/22 21:16:44
config.json json 6,171 2018/5/13 2:24:16
utils 2020/2/26 21:36:37
D:\engineercms
名称(N) 扩... 大小(Z)
database
log
views
static
controllers
swagger
routers
models
.git (h)
conf
app.conf conf
rbac_model.conf conf
config.json json

第二部 engineercms自身文件的修改

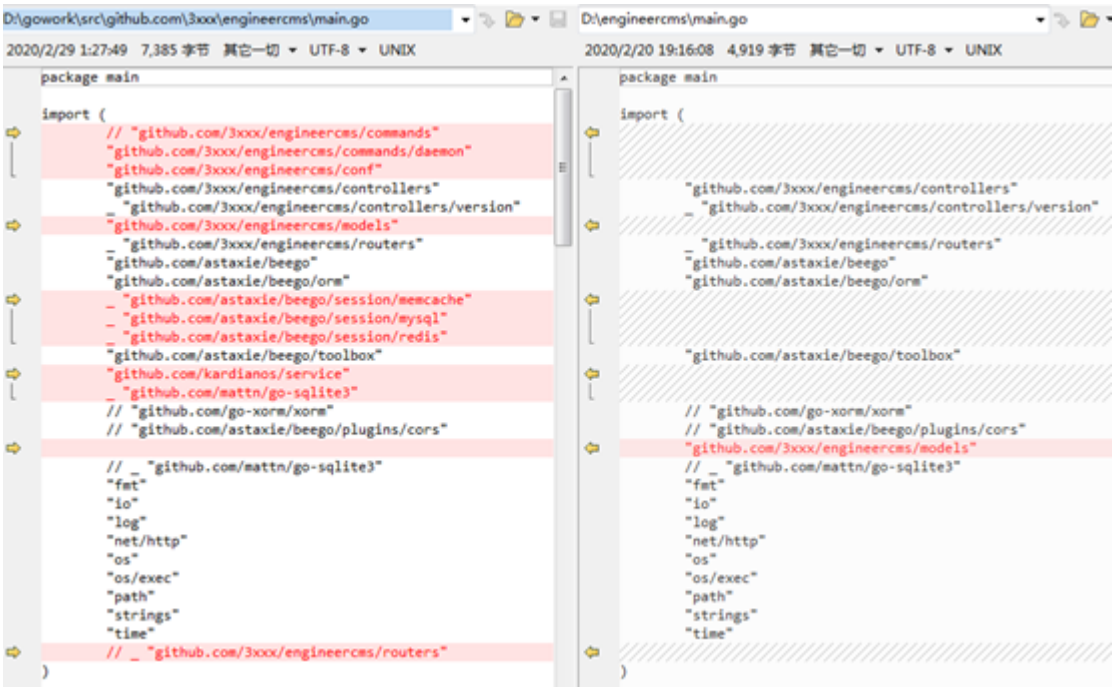
因为mindoc的地址都是github.com/lifei6671，所以都要换成github.com/3xxx我项目的名字。这里一个小插曲，我后来看bookstack，这个项目也是直接这样换的。看来对待go的项目整合，只有这种办法么？

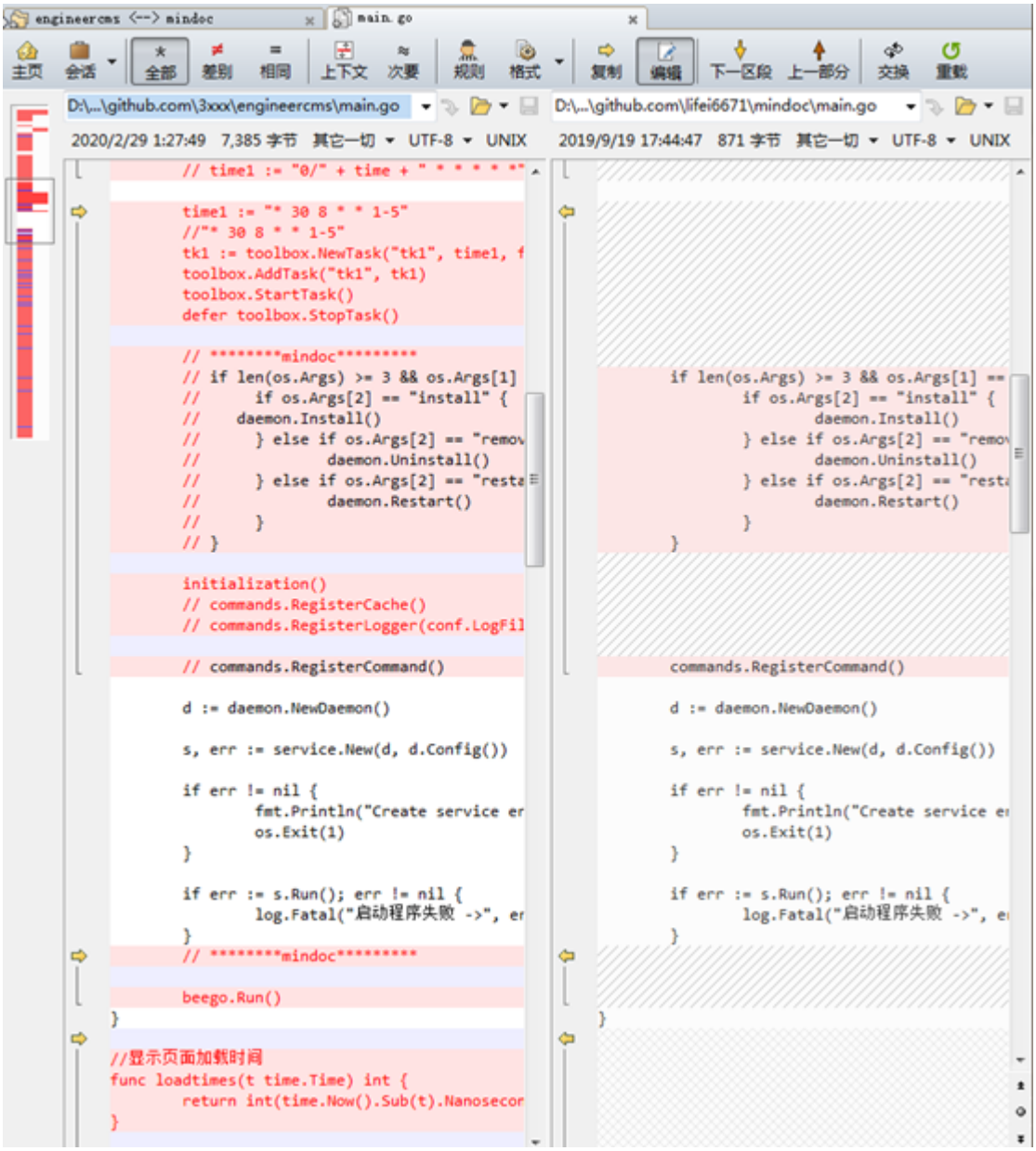
2 文件对比

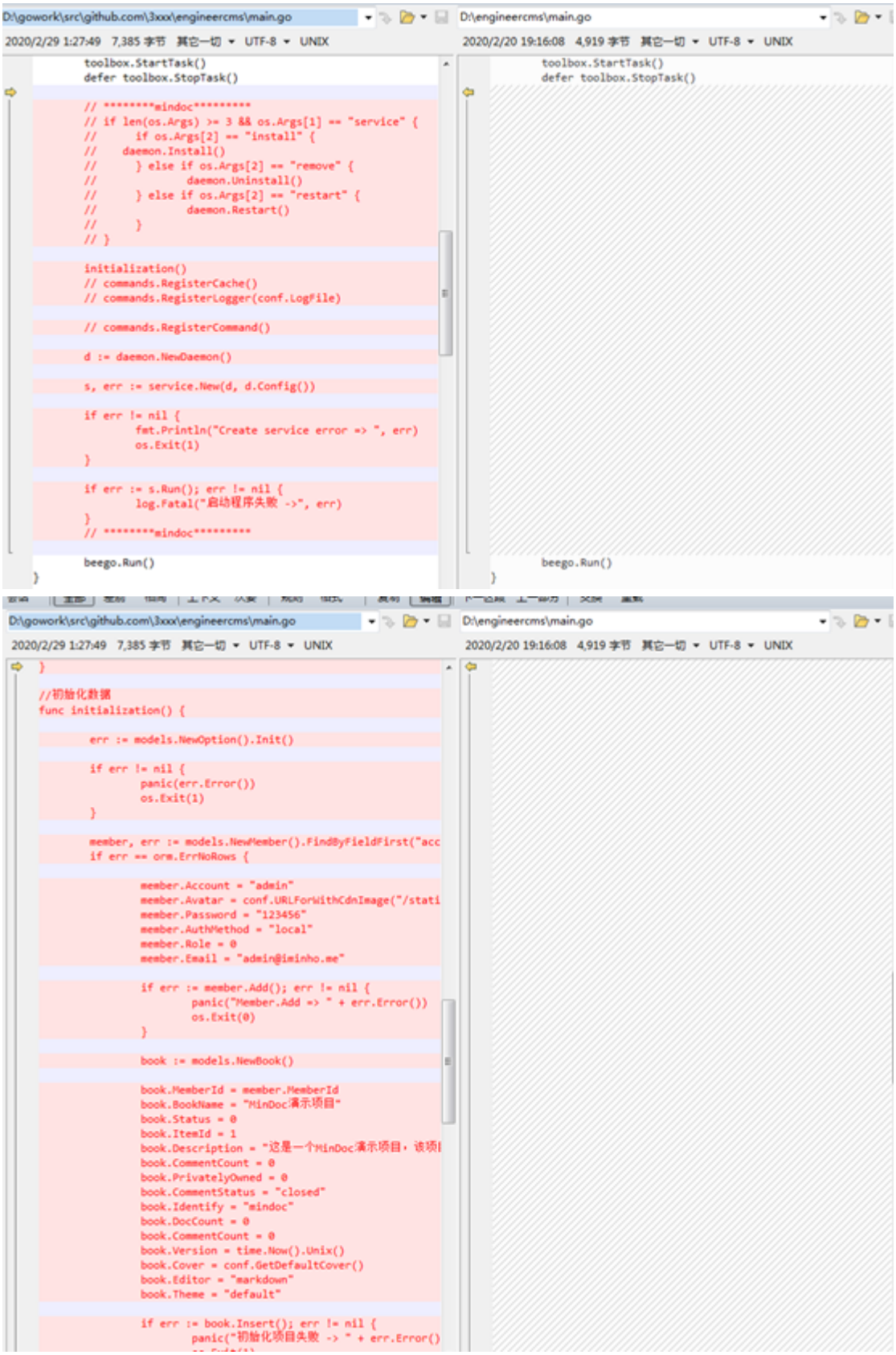
2.1 app.conf

#########引入mindoc配置########
include “app2.conf”

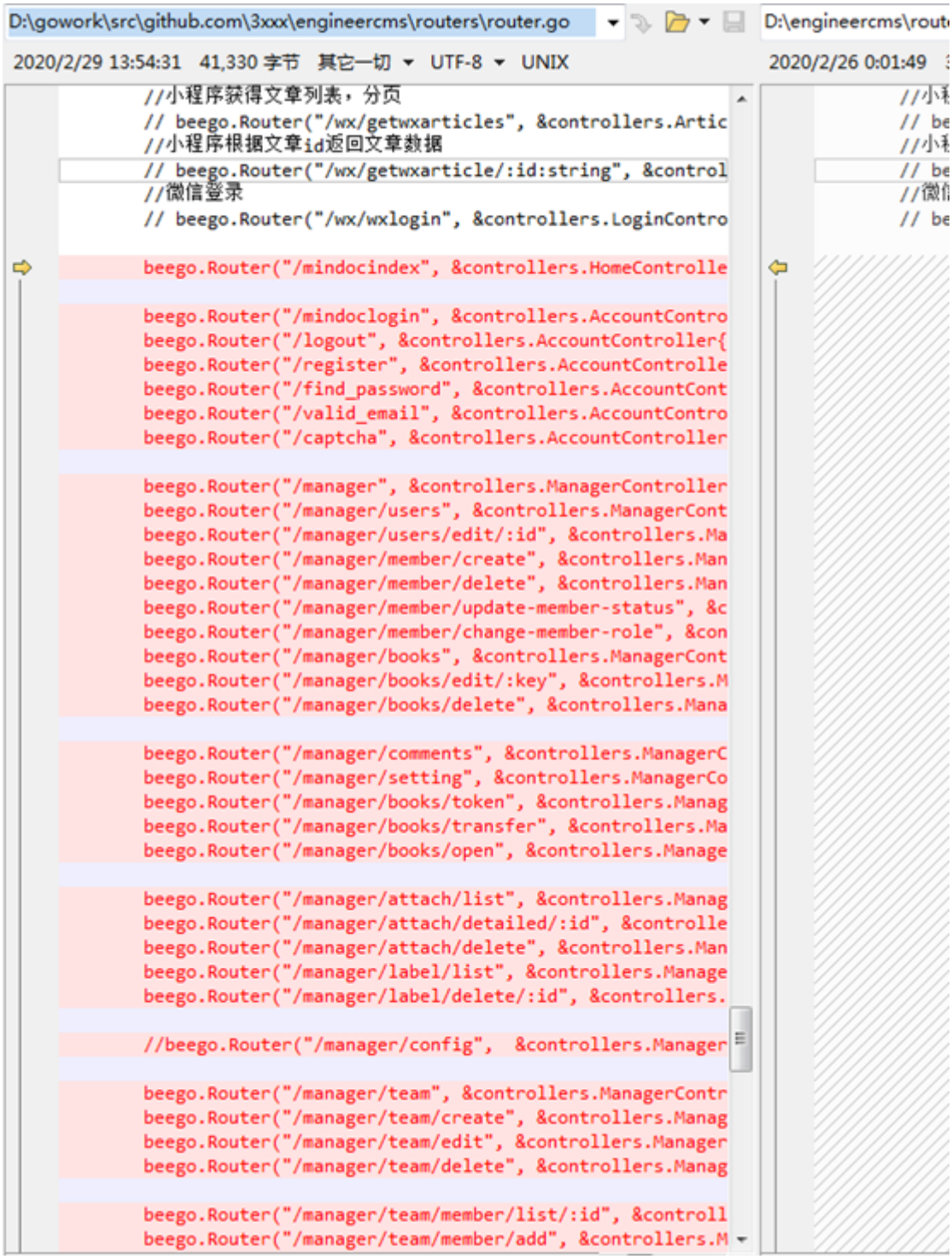
2.2 main


D:\gowork\src\github.com\3xxx\engineercms\main.go
2020/2/29 1:27:49 7,385 字节 其它一切 UTF-8 UNIX
D:\engineercms\main.go
2020/2/20 19:16:08 4,919 字节 其它一切 UTF-8 UNIX

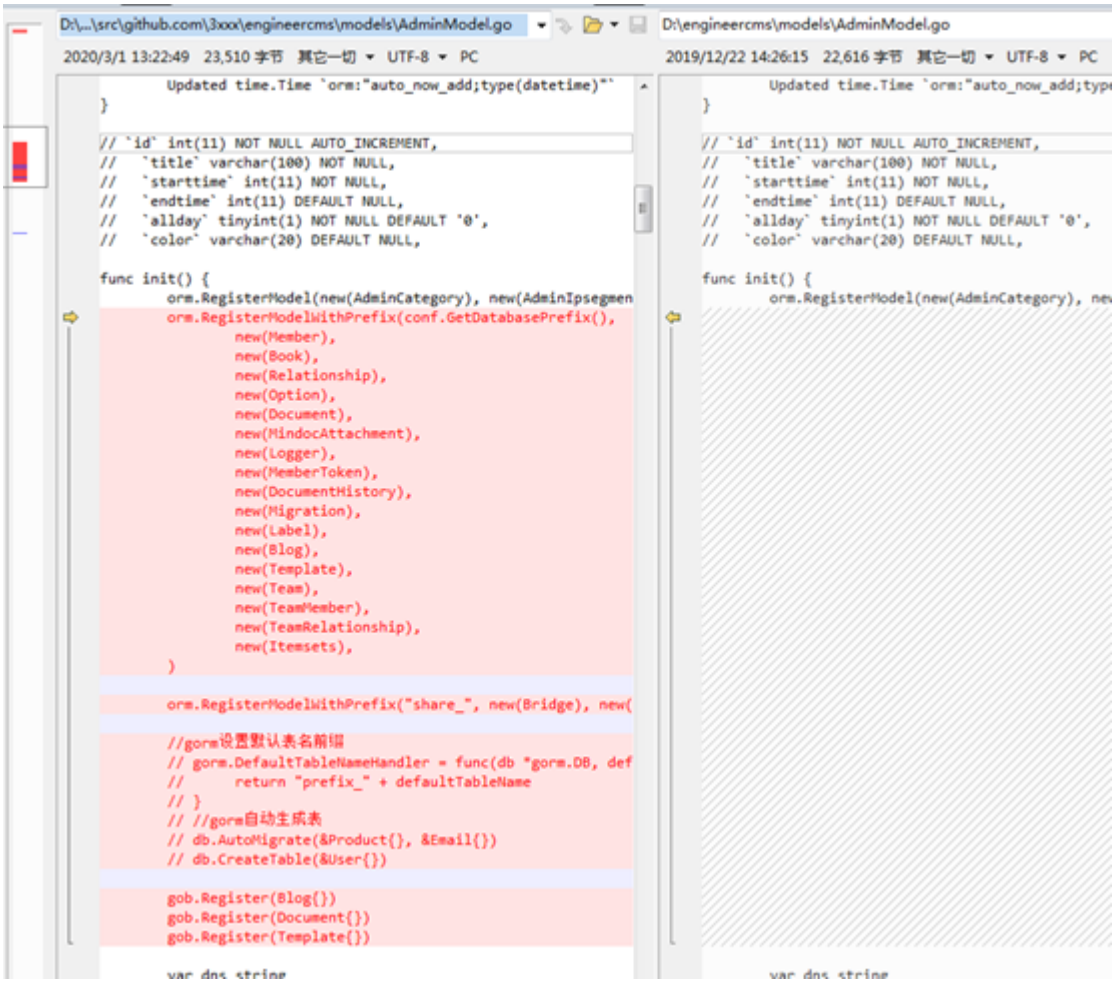

engineercms <--> mindoc
main.go
主页
会话
全部
差别
相同
上下文
次要
规则
格式
复制
编辑
下一区段
上一部分
交换
重载
D:\...\github.com\3xxx\engineercms\main.go
2020/2/29 1:27:49 7,385 字节 其它一切 UTF-8 UNIX
D:\...\github.com\lifei6671\mindoc\main.go
2019/9/19 17:44:47 871 字节 其它一切 UTF-8 UNIX
// time1 := "0/" + time + " * * * * *"
time1 := "* 30 8 * * 1-5"
//"* 30 8 * * 1-5"
tk1 := toolbox.NewTask("tk1", time1, f
toolbox.AddTask("tk1", tk1)
toolbox.StartTask()
defer toolbox.StopTask()
// ********mindoc*********
// if len(os.Args) >= 3 && os.Args[1]
// if os.Args[2] == "install" {
// daemon.Install()
// } else if os.Args[2] == "remov
// daemon.Uninstall()
// } else if os.Args[2] == "resta
// daemon.Restart()
// }
// }
initialization()
// commands.RegisterCache()
// commands.RegisterLogger(conf.LogFil
// commands.RegisterCommand()
d := daemon.NewDaemon()
s, err := service.New(d, d.Config())
if err != nil {
fmt.Println("Create service er
os.Exit(1)
}
if err := s.Run(); err != nil {
log.Fatal("启动程序失败 ->", er
}
// ********mindoc*********
beego.Run()
}
//显示页面加载时间
func loadtimes(t time.Time) int {
return int(time.Now().Sub(t).Nanosecon
}
if len(os.Args) >= 3 && os.Args[1] ==
if os.Args[2] == "install" {
daemon.Install()
} else if os.Args[2] == "remo
daemon.Uninstall()
} else if os.Args[2] == "resta
daemon.Restart()
}
}
commands.RegisterCommand()
d := daemon.NewDaemon()
s, err := service.New(d, d.Config())
if err != nil {
fmt.Println("Create service e
os.Exit(1)
}
if err := s.Run(); err != nil {
log.Fatal("启动程序失败 ->", e
}
}


D:\gowork\src\github.com\3xxx\engineercms\main.go
2020/2/29 1:27:49 7,385 字节 其它一切 UTF-8 UNIX
D:\engineercms\main.go
2020/2/20 19:16:08 4,919 字节 其它一切 UTF-8 UNIX
toolbox.StartTask()
defer toolbox.StopTask()
// ********mindoc*********
// if len(os.Args) >= 3 && os.Args[1] == "service" {
// if os.Args[2] == "install" {
// daemon.Install()
// } else if os.Args[2] == "remove" {
// daemon.Uninstall()
// } else if os.Args[2] == "restart" {
// daemon.Restart()
// }
// }
initialization()
// commands.RegisterCache()
// commands.RegisterLogger(conf.LogFile)
// commands.RegisterCommand()
d := daemon.NewDaemon()
s, err := service.New(d, d.Config())
if err != nil {
fmt.Println("Create service error => ", err)
os.Exit(1)
}
if err := s.Run(); err != nil {
log.Fatal("启动程序失败 ->", err)
}
// ********mindoc*********
beego.Run()
}
beego.Run()
}
//初始化数据
func initialization() {
err := models.NewOption().Init()
if err != nil {
panic(err.Error())
os.Exit(1)
}
member, err := models.NewMember().FindByFieldFirst("acc
if err == orm.ErrNoRows {
member.Account = "admin"
member.Avatar = conf.URLForWithCdnImage("/stati
member.Password = "123456"
member.AuthMethod = "local"
member.Role = 0
member.Email = "admin@iminho.me"
if err := member.Add(); err != nil {
panic("Member.Add => " + err.Error())
os.Exit(0)
}
book := models.NewBook()
book.MemberId = member.MemberId
book.BookName = "MinDoc演示项目"
book.Status = 0
book.ItemId = 1
book.Description = "这是一个MinDoc演示项目，该项目
book.CommentCount = 0
book.PrivatelyOwned = 0
book.CommentStatus = "closed"
book.Identify = "mindoc"
book.DocCount = 0
book.CommentCount = 0
book.Version = time.Now().Unix()
book.Cover = conf.GetDefaultCover()
book.Editor = "markdown"
book.Theme = "default"
if err := book.Insert(); err != nil {
panic("初始化项目失败 -> " + err.Error()

2.3 route



2.4 adminmodel


D:\...\src\github.com\3xxx\engineercms\models\AdminModel.go
2020/3/1 13:22:49 23,510 字节 其它一切 UTF-8 PC
D:\engineercms\models\AdminModel.go
2019/12/22 14:26:15 22,616 字节 其它一切 UTF-8 PC
Updated time.Time `orm:"auto_now_add;type(datetime)"`
}
// `id` int(11) NOT NULL AUTO_INCREMENT,
// `title` varchar(100) NOT NULL,
// `starttime` int(11) NOT NULL,
// `endtime` int(11) DEFAULT NULL,
// `allday` tinyint(1) NOT NULL DEFAULT '0',
// `color` varchar(20) DEFAULT NULL,
func init() {
orm.RegisterModel(new(AdminCategory), new(AdminIpsegmen
orm.RegisterModelWithPrefix(conf.GetDatabasePrefix(),
new(Member),
new(Book),
new(Relationship),
new(Option),
new(Document),
new(MindocAttachment),
new(Logger),
new(MemberToken),
new(DocumentHistory),
new(Migration),
new(Label),
new(Blog),
new(Template),
new(Team),
new(TeamMember),
new(TeamRelationship),
new(Itemsets),
)
orm.RegisterModelWithPrefix("share_", new(Bridge), new(
//gorm设置默认表名前缀
// gorm.DefaultTableNameHandler = func(db *gorm.DB, def
// return "prefix_" + defaultTableName
// }
// //gorm自动生成表
// db.AutoMigrate(&Product{}, &Email{})
// db.CreateTable(&User{})
gob.Register(Blog{})
gob.Register(Document{})
gob.Register(Template{})
var dns string
Updated time.Time `orm:"auto_now_add;type
}
// `id` int(11) NOT NULL AUTO_INCREMENT,
// `title` varchar(100) NOT NULL,
// `starttime` int(11) NOT NULL,
// `endtime` int(11) DEFAULT NULL,
// `allday` tinyint(1) NOT NULL DEFAULT '0',
// `color` varchar(20) DEFAULT NULL,
func init() {
orm.RegisterModel(new(AdminCategory), new
var dns string

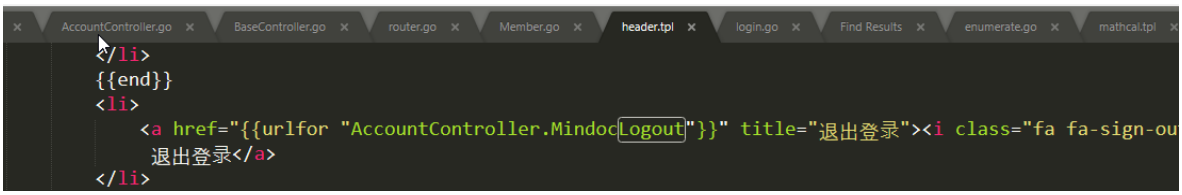
第三步 mindoc文件的修改

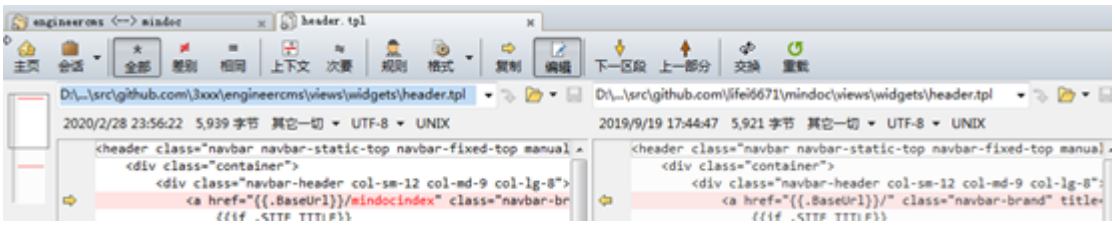
首先是将golang升级到最新，然后是下载各种依赖的包。

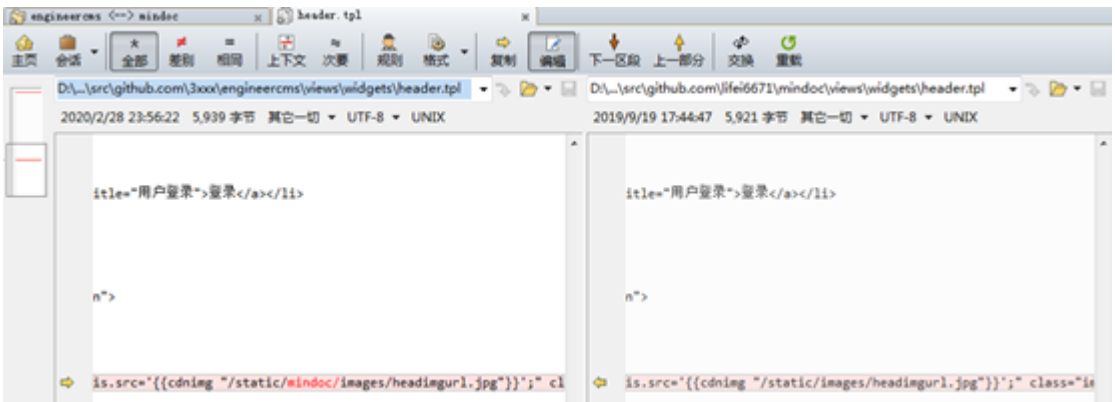
3 mindoc修改

3.1 header.tpl

因为logout与ecms冲突，所以修改mindoc的logout为mindoclogout
直接改header.tpl里的urlfor(位于 `conf—enumerate.go`)后面的
`urlfor "AccountController.Logout"` 为
`urlfor "AccountController.MindocLogout"` ,
再修改router中的
`beego.Router("/logout", &controllers.AccountController{}, "*:Logout")` 为
`beego.Router("/mindoclogout", &controllers.AccountController{}, "*:Logout")`



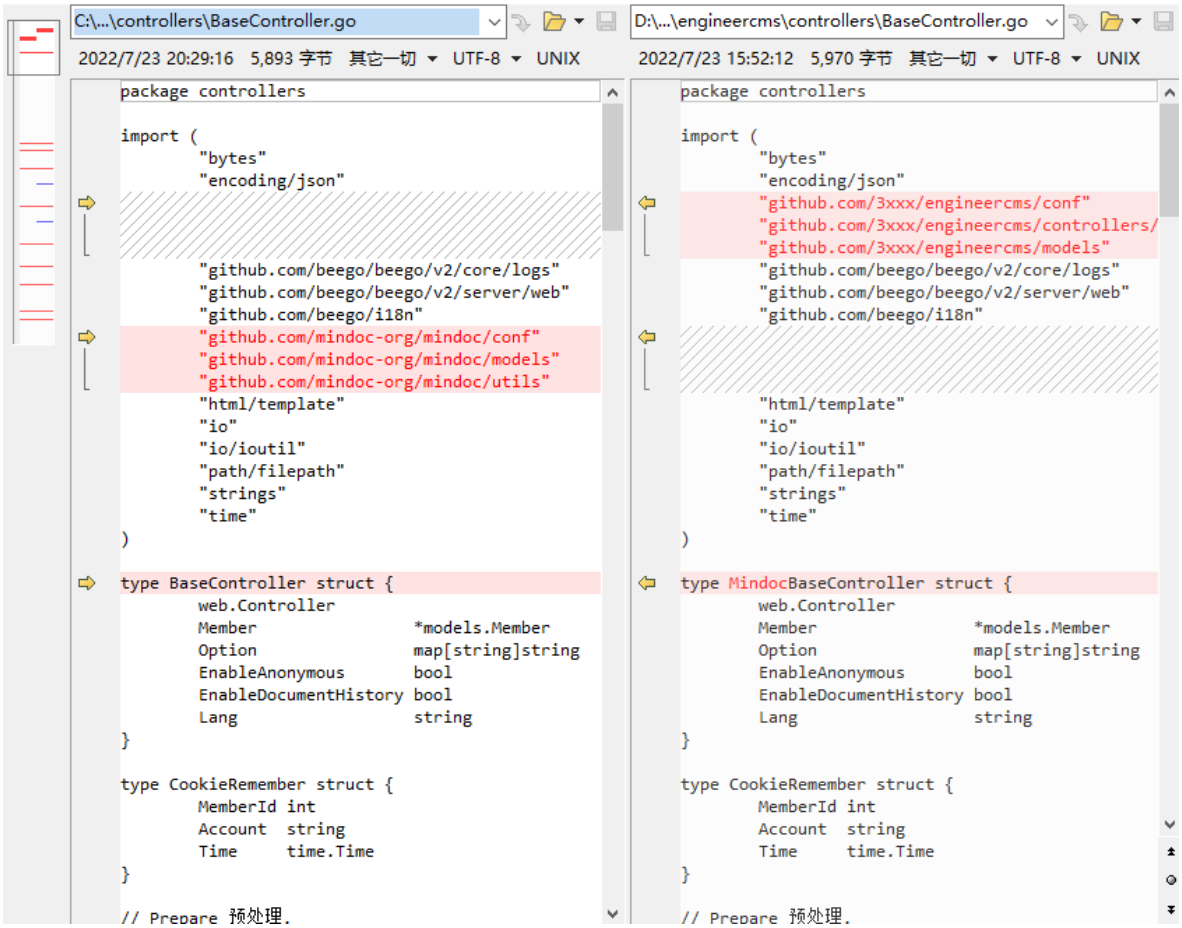




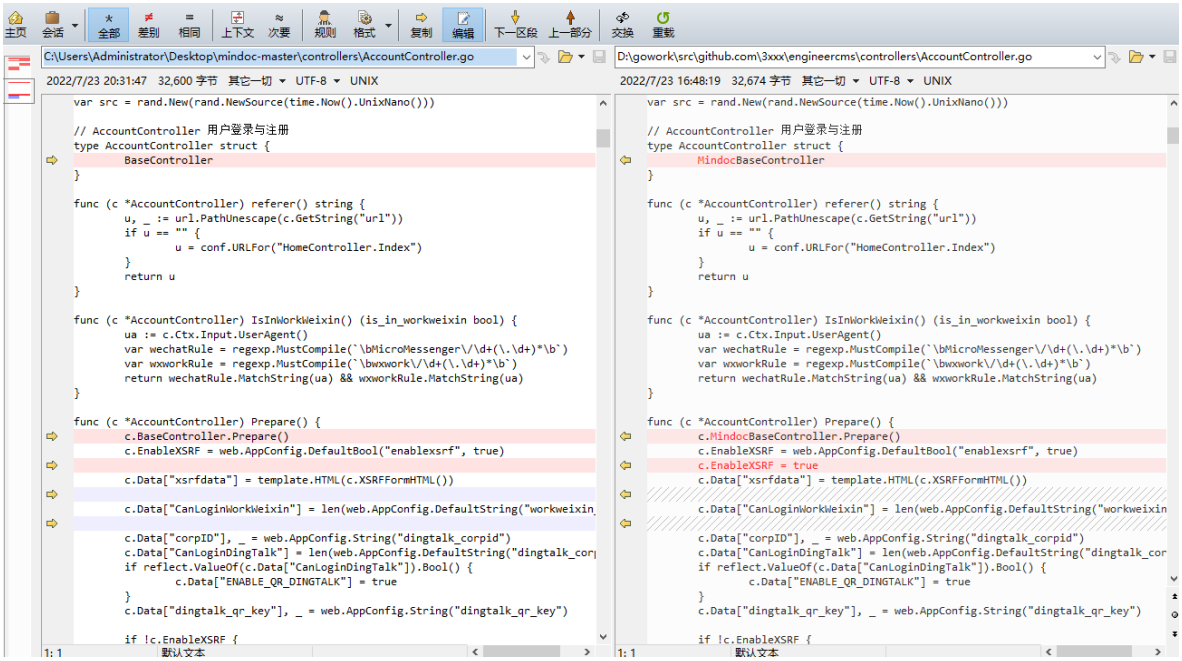
其余view都是修改增加mindoc这个静态资源的路径

3.2 BaseController.go修改

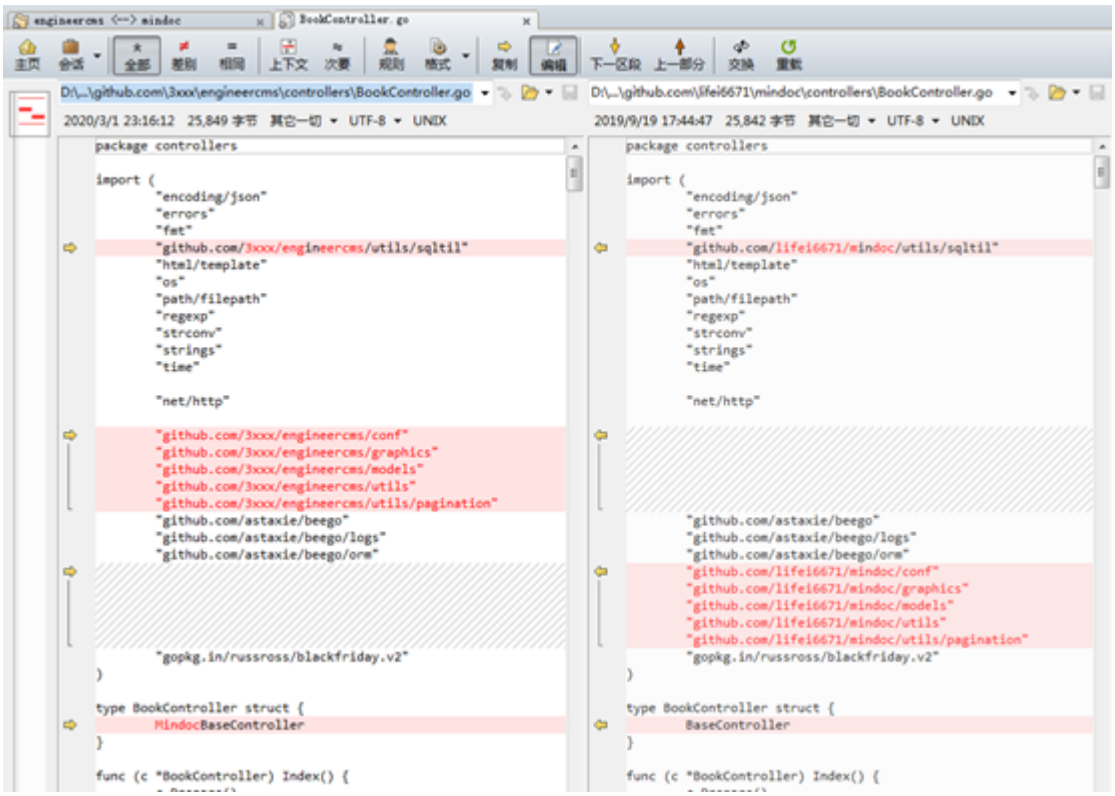
BaseController改为MindocBaseController



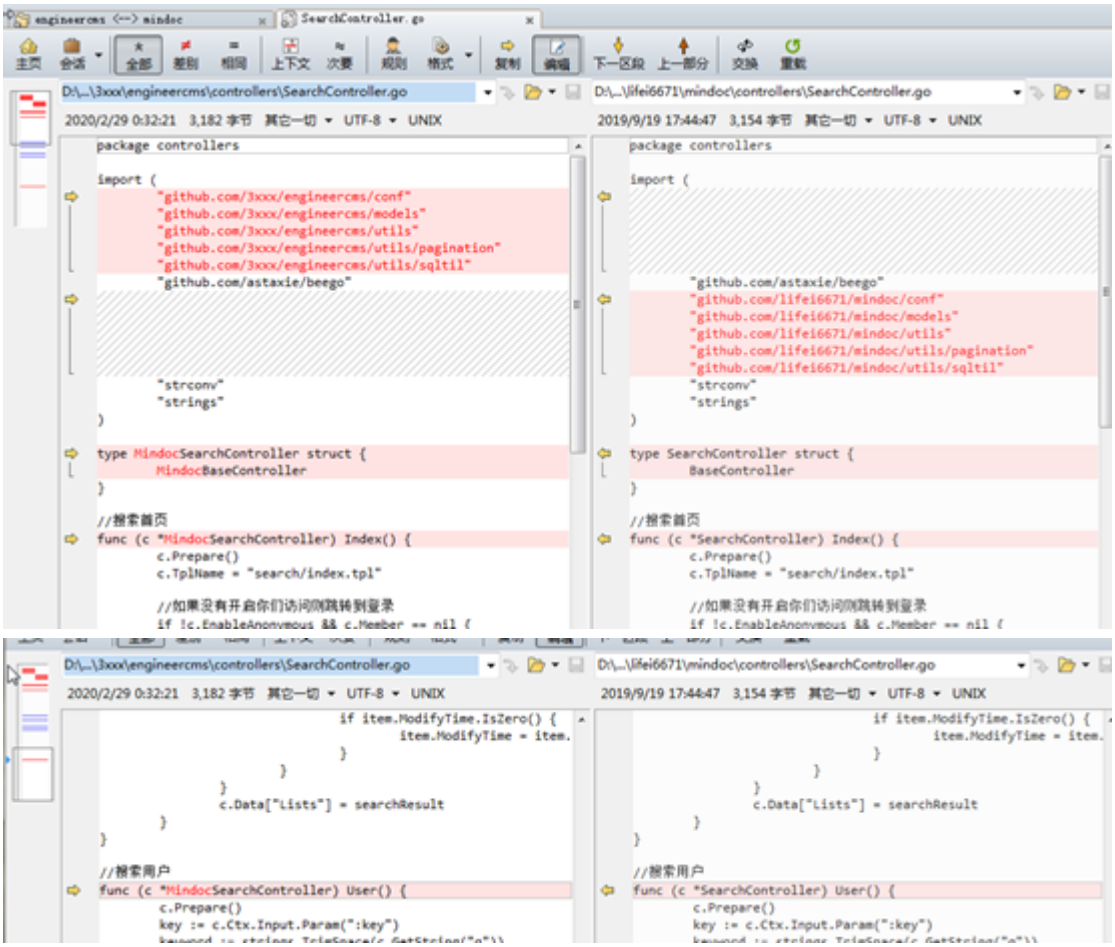
其他所有引用BaseController的也要改到，如AccountController。



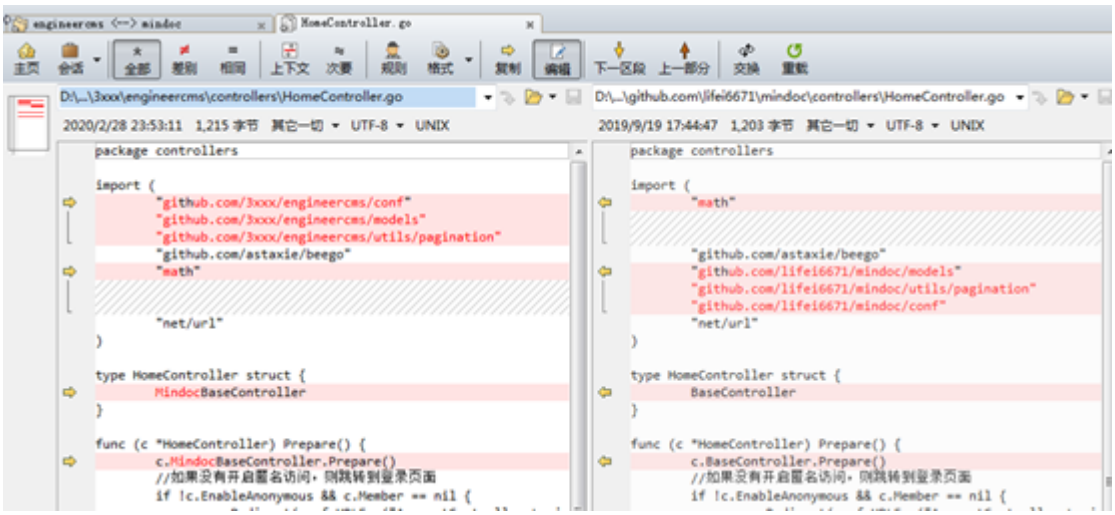
3.2 bookcontroller.go



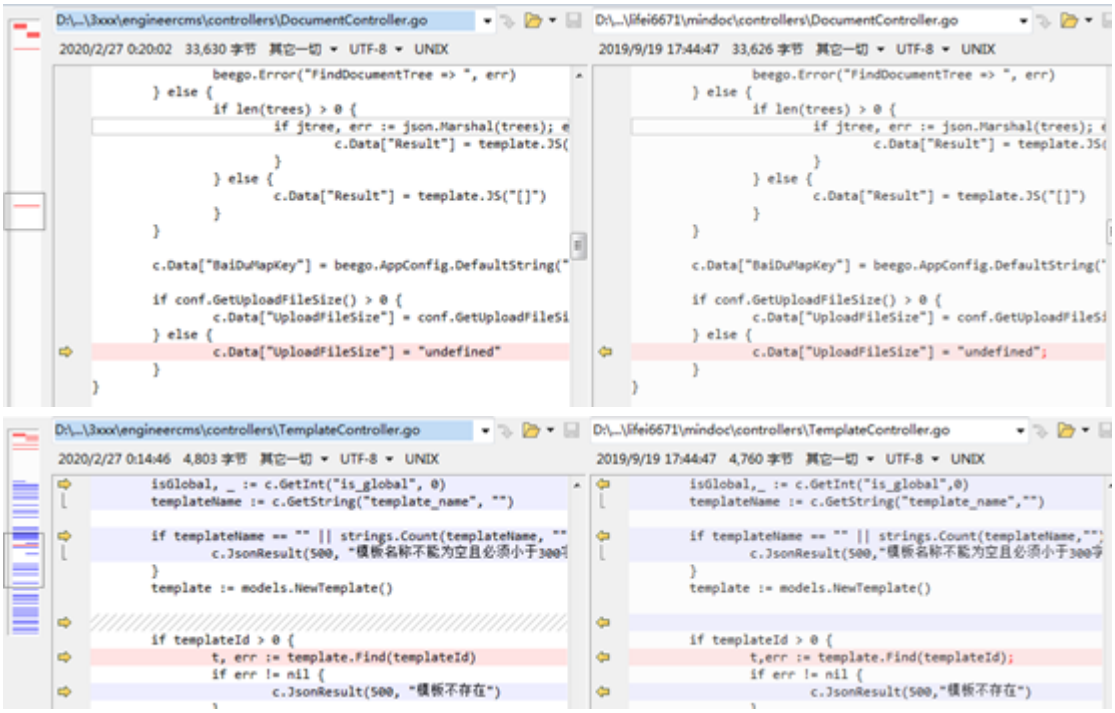
3.3 searchcontroller



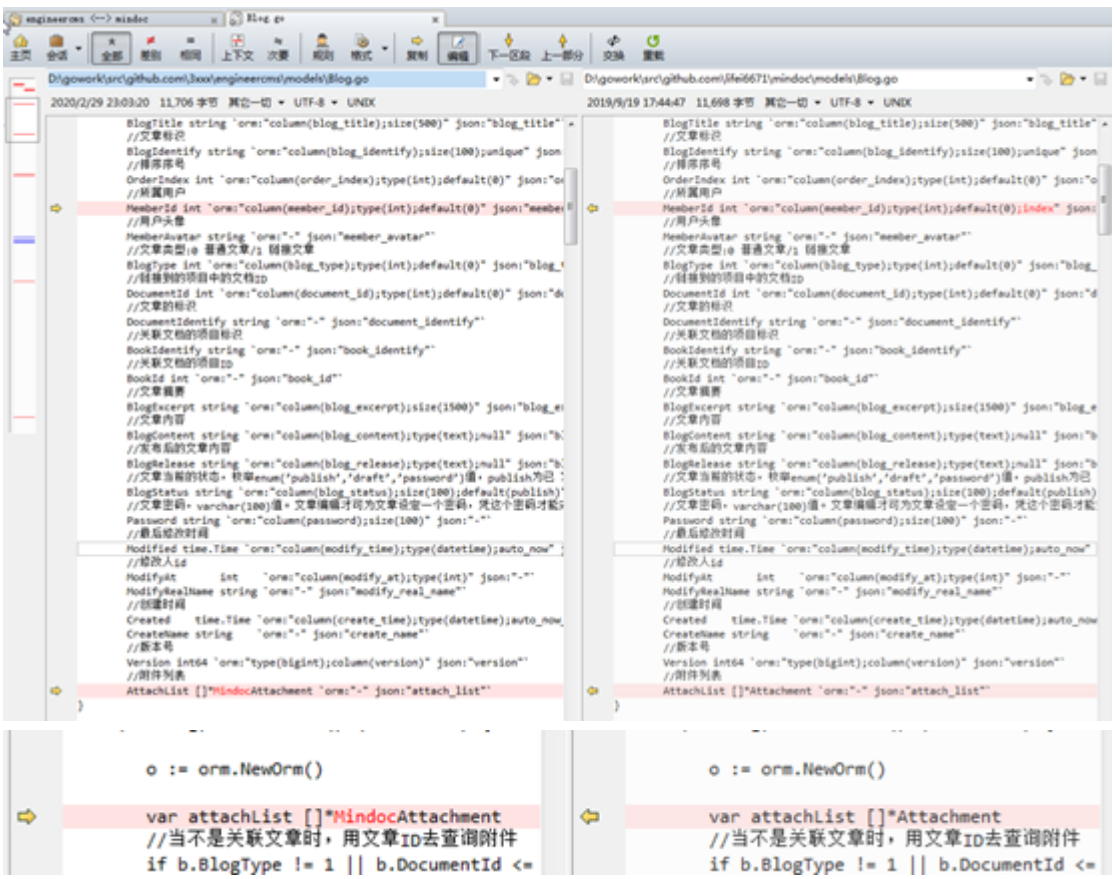
3.4 homecontroller.go



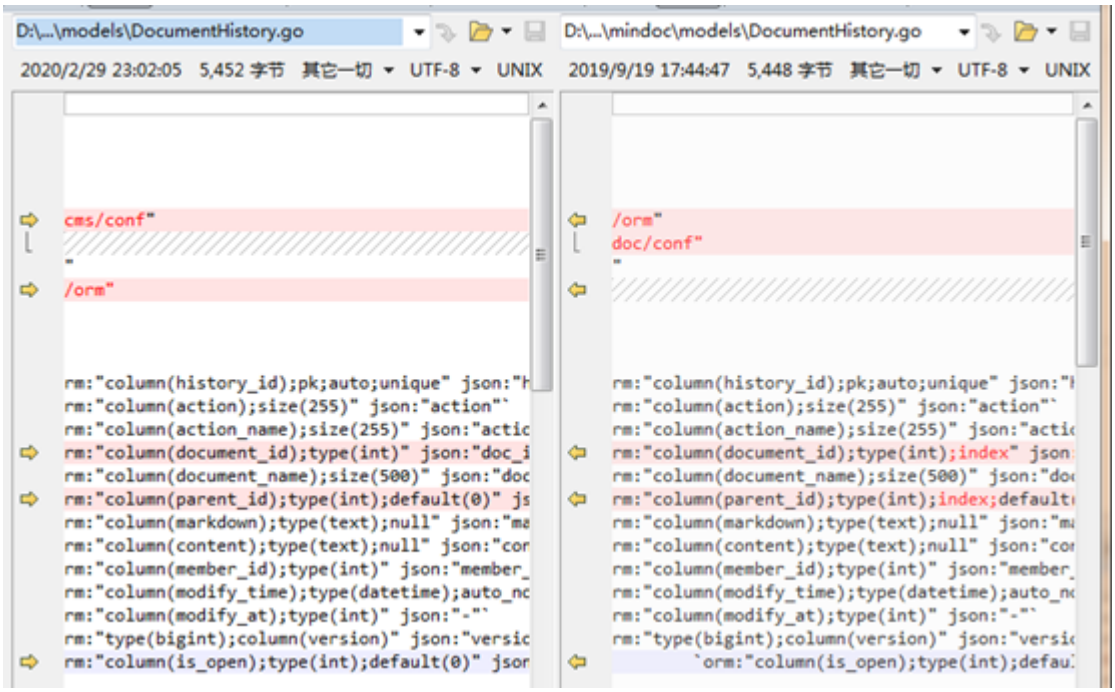
其余都是MindocBaseController


D:\...\3xxx\engineercms\controllers\DocumentController.go
2020/2/27 0:20:02 33,630 字节 其它一切 UTF-8 UNIX
D:\...\lifei6671\mindoc\controllers\DocumentController.go
2019/9/19 17:44:47 33,626 字节 其它一切 UTF-8 UNIX
beego.Error("FindDocumentTree => ", err)
} else {
if len(trees) > 0 {
if jtree, err := json.Marshal(trees); e
c.Data["Result"] = template.JS(
}
} else {
c.Data["Result"] = template.JS("[]")
}
}
c.Data["BaiDuMapKey"] = beego.AppConfig.DefaultString("
if conf.GetUploadFileSize() > 0 {
c.Data["UploadFileSize"] = conf.GetUploadFileSi
} else {
c.Data["UploadFileSize"] = "undefined"
}
}
D:\...\3xxx\engineercms\controllers\TemplateController.go
2020/2/27 0:14:46 4,803 字节 其它一切 UTF-8 UNIX
D:\...\lifei6671\mindoc\controllers\TemplateController.go
2019/9/19 17:44:47 4,760 字节 其它一切 UTF-8 UNIX
isGlobal, _ := c.GetInt("is_global", 0)
templateName := c.GetString("template_name", "")
if templateName == "" || strings.Count(templateName, ""
c.JsonResult(500, "模板名称不能为空且必须小于300字
}
template := models.NewTemplate()
if templateId > 0 {
t, err := template.Find(templateId)
if err != nil {
c.JsonResult(500, "模板不存在")

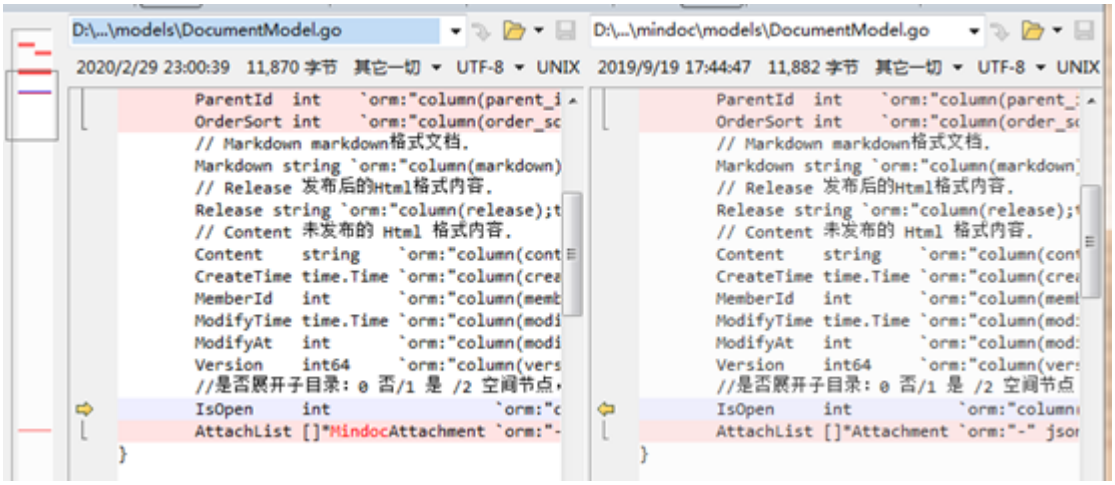
3.5 blog.go

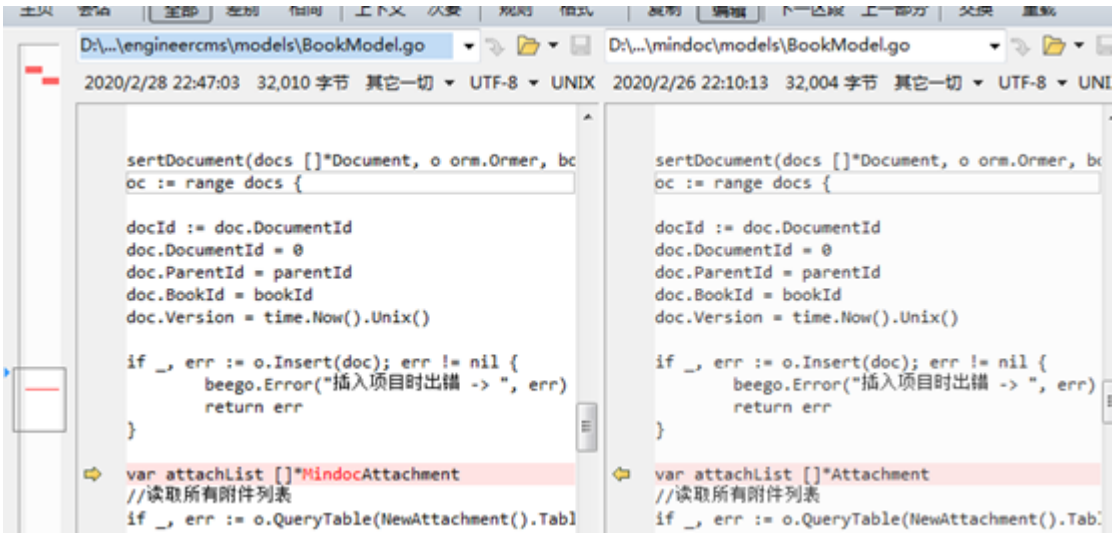

D:\gowork\src\github.com\3xxx\engineercms\models\Blog.go
2020/2/29 23:03:20 11,706 字节 其它一切 UTF-8 UNIX
D:\gowork\src\github.com\lifei6671\mindoc\models\Blog.go
2019/9/19 17:44:47 11,698 字节 其它一切 UTF-8 UNIX
o := orm.NewOrm()
var attachList []*MindocAttachment
//当不是关联文章时，用文章ID去查询附件
if b.BlogType != 1 || b.DocumentId <=
o := orm.NewOrm()
var attachList []*Attachment
//当不是关联文章时，用文章ID去查询附件
if b.BlogType != 1 || b.DocumentId <=

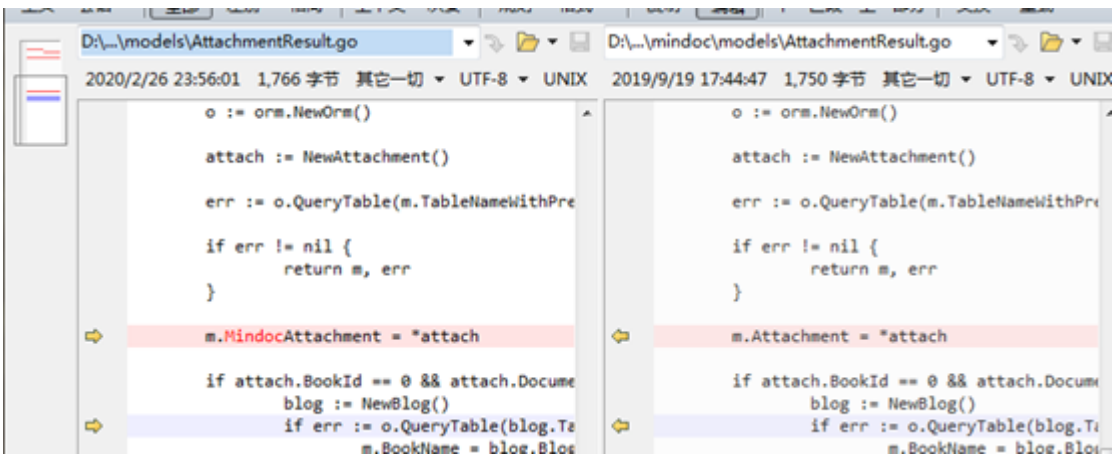
其他是去掉index，否则无法自动建表。

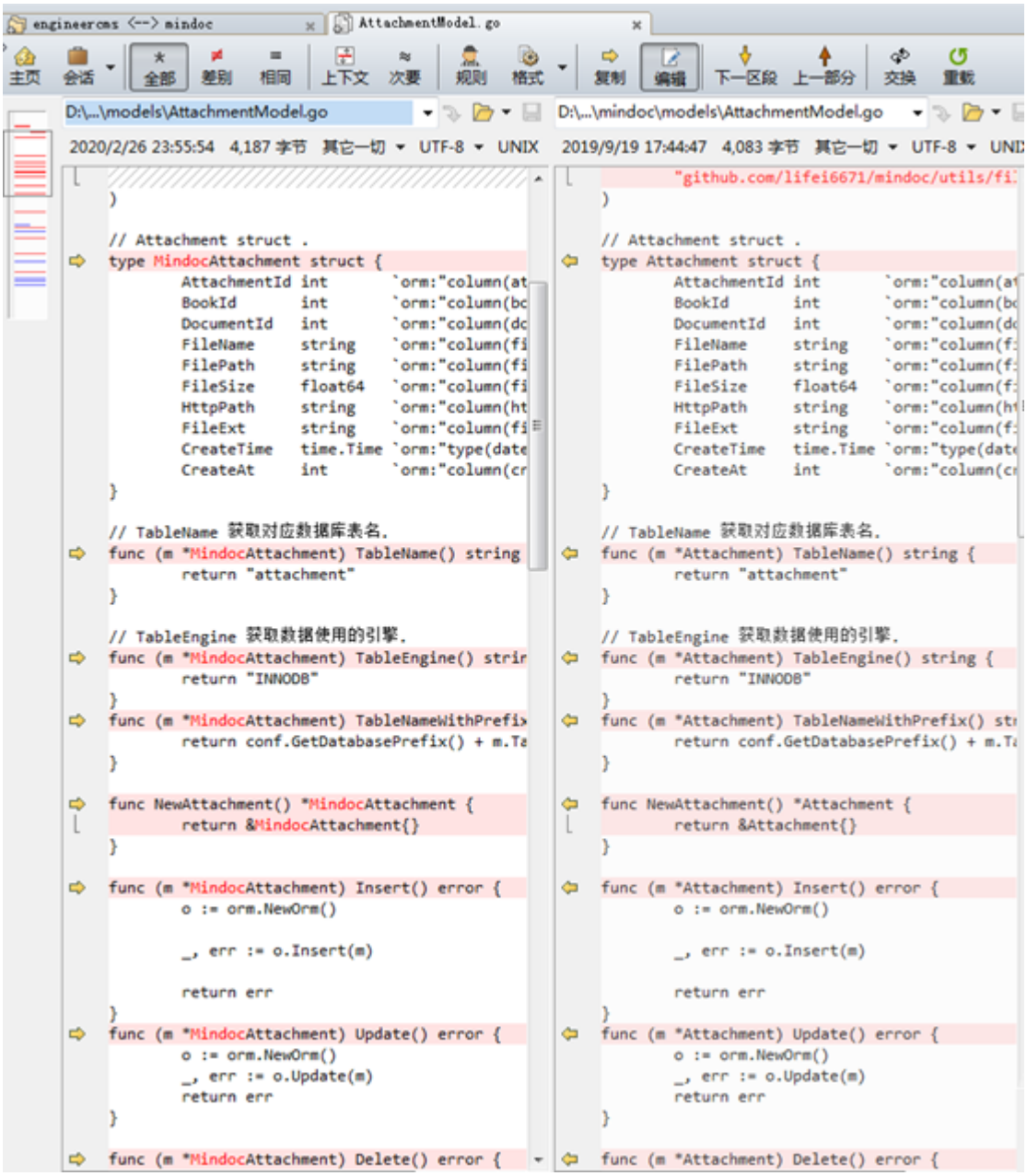


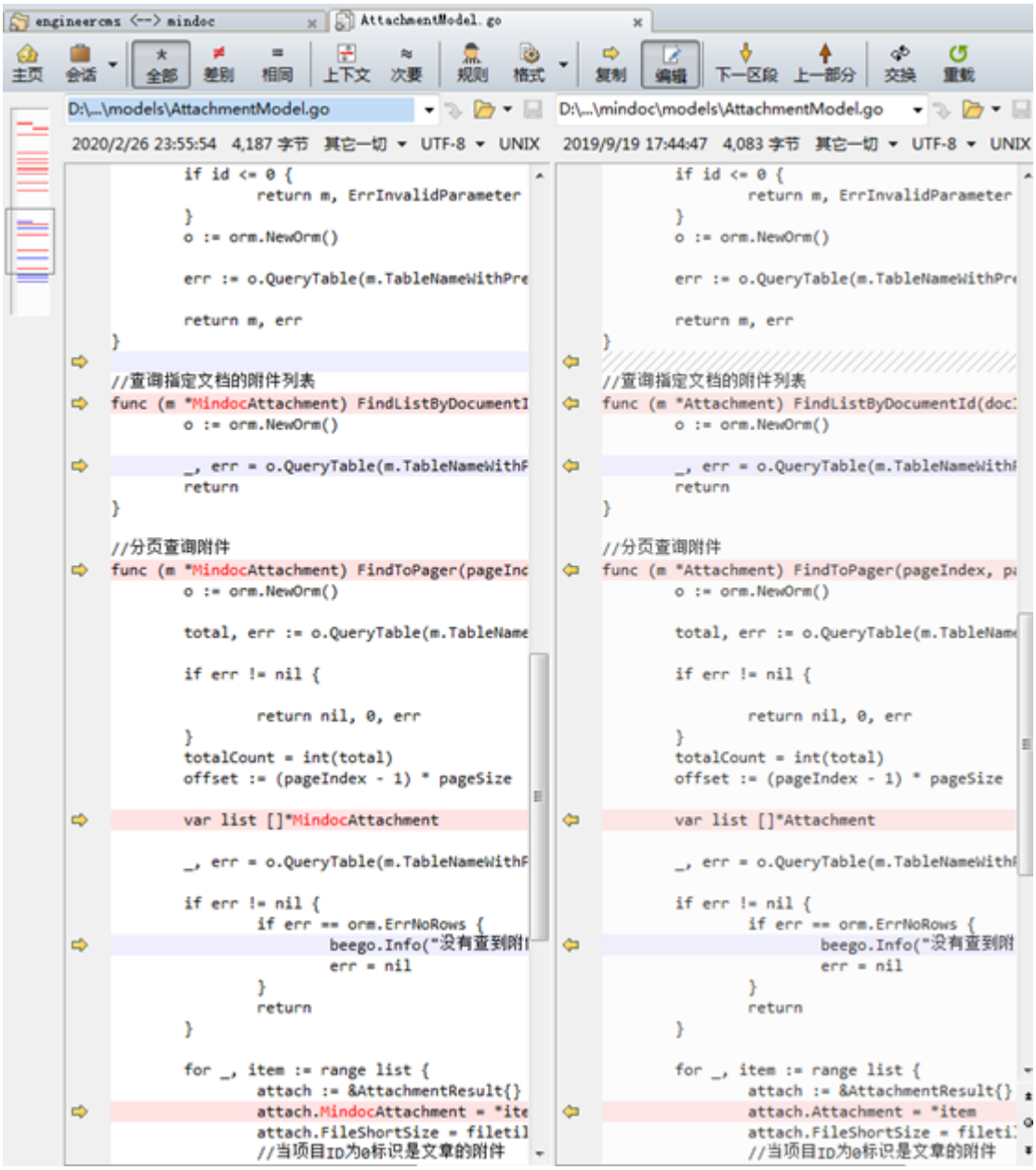
3.6 attachment的修改


D:\...\models\DocumentModel.go
2020/2/29 23:00:39 11,870 字节 其它一切 UTF-8 UNIX
D:\...\mindoc\models\DocumentModel.go
2019/9/19 17:44:47 11,882 字节 其它一切 UTF-8 UNIX
ParentId int `orm:"column(parent_i
OrderSort int `orm:"column(order_sc
// Markdown markdown格式文档.
Markdown string `orm:"column(markdown)
// Release 发布后的Html格式内容.
Release string `orm:"column(release);t
// Content 未发布的 Html 格式内容.
Content string `orm:"column(cont
CreateTime time.Time `orm:"column(crea
MemberId int `orm:"column(memb
ModifyTime time.Time `orm:"column(modi
ModifyAt int `orm:"column(modi
Version int64 `orm:"column(vers
//是否展开子目录: 0 否/1 是 /2 空阀节点
IsOpen int `orm:"c
AttachList []*MindocAttachment `orm:"-
}
ParentId int `orm:"column(parent_
OrderSort int `orm:"column(order_sc
// Markdown markdown格式文档.
Markdown string `orm:"column(markdown)
// Release 发布后的Html格式内容.
Release string `orm:"column(release);t
// Content 未发布的 Html 格式内容.
Content string `orm:"column(cont
CreateTime time.Time `orm:"column(crea
MemberId int `orm:"column(memb
ModifyTime time.Time `orm:"column(mod
ModifyAt int `orm:"column(mod
Version int64 `orm:"column(vers
//是否展开子目录: 0 否/1 是 /2 空阀节点
IsOpen int `orm:"column
AttachList []*Attachment `orm:"-" jso
}

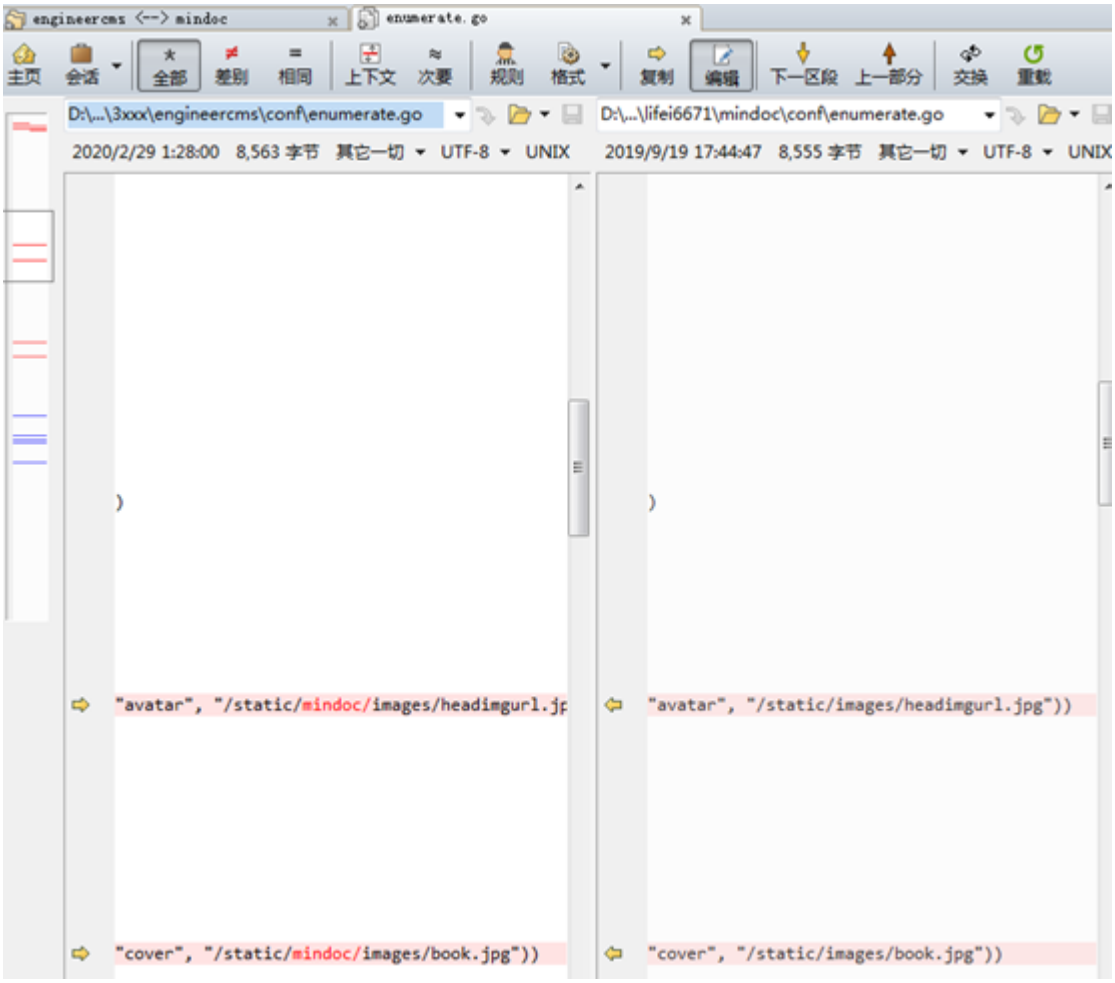

D:\...\engineercms\models\BookModel.go
2020/2/28 22:47:03 32,010 字节 其它一切 UTF-8 UNIX
D:\...\mindoc\models\BookModel.go
2020/2/26 22:10:13 32,004 字节 其它一切 UTF-8 UNI
sertDocument(docs []*Document, o orm.Ormer, bc
oc := range docs {
docId := doc.DocumentId
doc.DocumentId = 0
doc.ParentId = parentId
doc.BookId = bookId
doc.Version = time.Now().Unix()
if _, err := o.Insert(doc); err != nil {
beego.Error("插入项目时出错 -> ", err)
return err
}
var attachList []*MindocAttachment
//读取所有附件列表
if _, err := o.QueryTable(NewAttachment().Tabl
sertDocument(docs []*Document, o orm.Ormer, bc
oc := range docs {
docId := doc.DocumentId
doc.DocumentId = 0
doc.ParentId = parentId
doc.BookId = bookId
doc.Version = time.Now().Unix()
if _, err := o.Insert(doc); err != nil {
beego.Error("插入项目时出错 -> ", err)
return err
}
var attachList []*Attachment
//读取所有附件列表
if _, err := o.QueryTable(NewAttachment().Tab


D:\...\models\AttachmentResult.go
2020/2/26 23:56:01 1,766 字节 其它一切 UTF-8 UNIX
D:\...\mindoc\models\AttachmentResult.go
2019/9/19 17:44:47 1,750 字节 其它一切 UTF-8 UNIX
o := orm.NewOrm()
attach := NewAttachment()
err := o.QueryTable(m.TableNameWithPre
if err != nil {
return m, err
}
m.MindocAttachment = *attach
if attach.BookId == 0 && attach.Docume
blog := NewBlog()
if err := o.QueryTable(blog.Ta
o := orm.NewOrm()
attach := NewAttachment()
err := o.QueryTable(m.TableNameWithPr
if err != nil {
return m, err
}
m.Attachment = *attach
if attach.BookId == 0 && attach.Docum
blog := NewBlog()
if err := o.QueryTable(blog.Ta

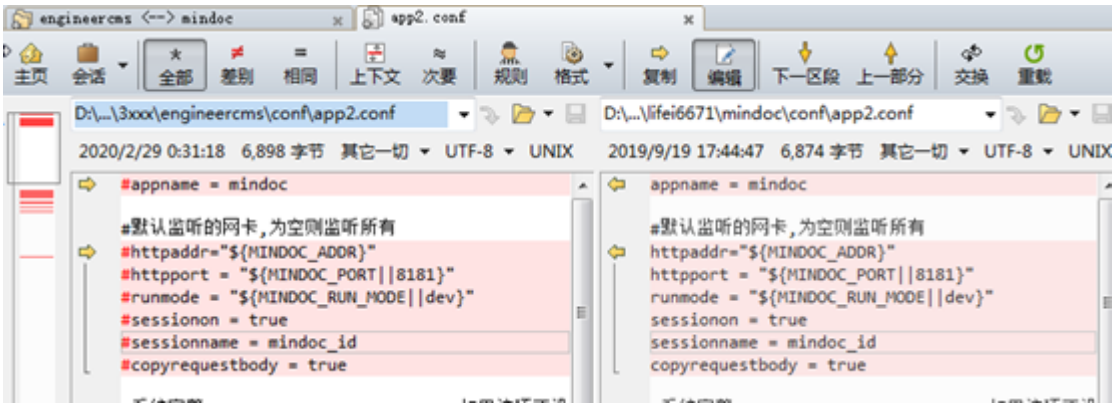

engineercms <--> mindoc
AttachmentModel.go
主页
会话
全部
差别
相同
上下文
次要
规则
格式
复制
编辑
下一区段
上一部分
交换
重载
D:\...\models\AttachmentModel.go
D:\...\mindoc\models\AttachmentModel.go
2020/2/26 23:55:54 4,187 字节 其它一切 UTF-8 UNIX
2019/9/19 17:44:47 4,083 字节 其它一切 UTF-8
// Attachment struct .
type MindocAttachment struct {
type Attachment struct {
// TableName 获取对应数据库表名.
// TableEngine 获取数据使用的引擎.
return "INNODB"
return "attachment"
func NewAttachment() *MindocAttachment {
func NewAttachment() *Attachment {
return &MindocAttachment{}
return &Attachment{}
func (m *MindocAttachment) Insert() error {
func (m *Attachment) Insert() error {
o := orm.NewOrm()
_, err := o.Insert(m)
return err
func (m *MindocAttachment) Update() error {
func (m *Attachment) Update() error {
_, err := o.Update(m)
func (m *MindocAttachment) Delete() error {
func (m *Attachment) Delete() error {



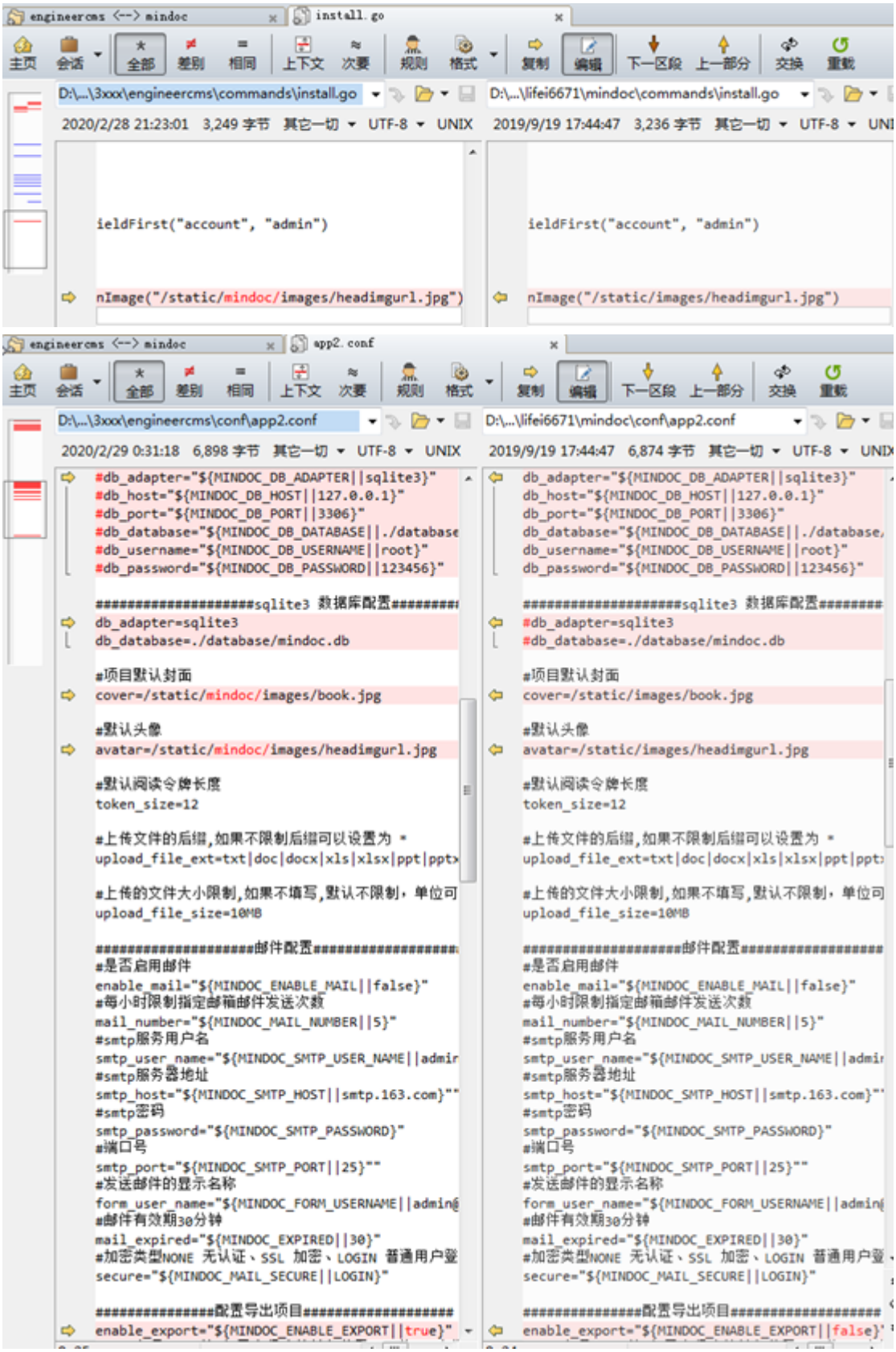
3.7 conf里enumerate.go



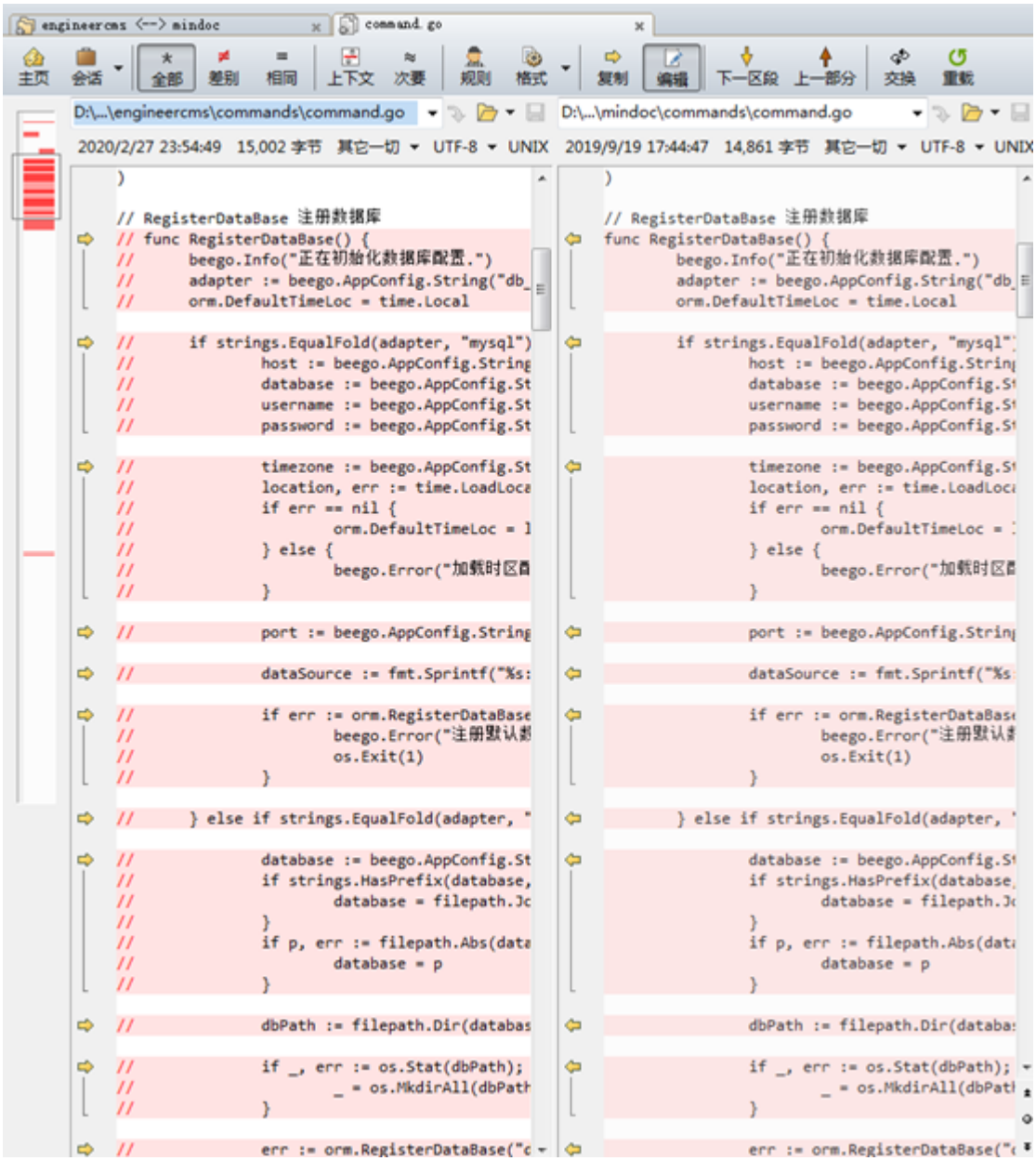
3.7 conf



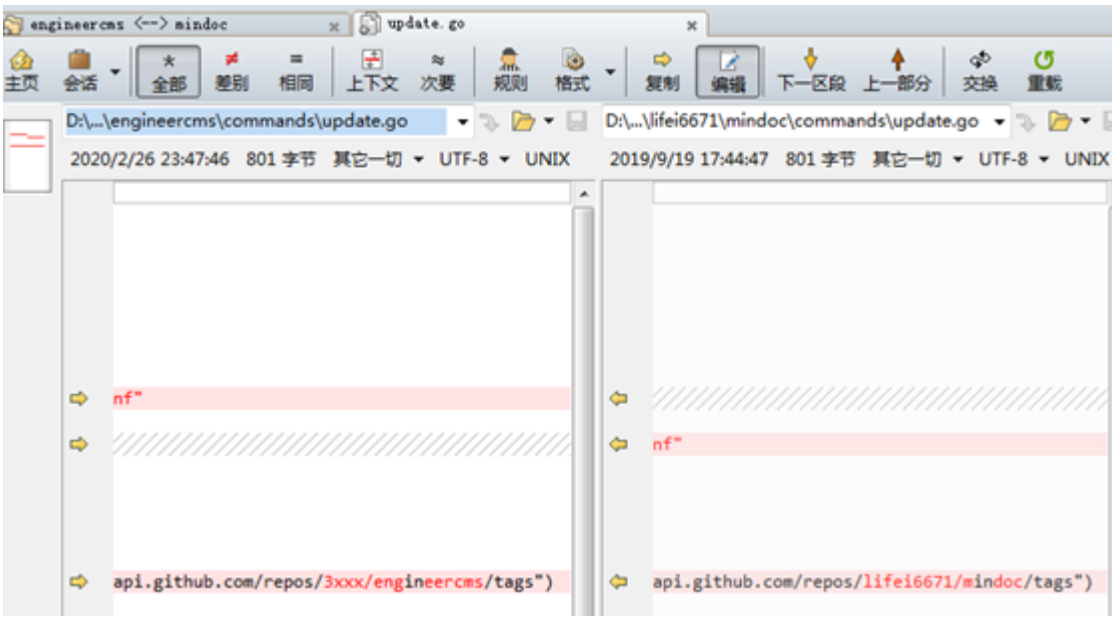
3.8 commands里的install.go



3.9 command.go


engineercms <--> mindoc
command.go
主页
会话
全部
差别
相同
上下文
次要
规则
格式
复制
编辑
下一区段
上一部分
交换
重载
D:\...\engineercms\commands\command.go
D:\...\mindoc\commands\command.go
2020/2/27 23:54:49 15,002 字节 其它一切 UTF-8 UNIX
2019/9/19 17:44:47 14,861 字节 其它一切 UTF-8 UNIX
// RegisterDataBase 注册数据库
// func RegisterDataBase() {
func RegisterDataBase() {
beego.Info("正在初始化数据库配置.")
adapter := beego.AppConfig.String("db_
orm.DefaultTimeLoc = time.Local
if strings.EqualFold(adapter, "mysql")
host := beego.AppConfig.String
database := beego.AppConfig.St
username := beego.AppConfig.St
password := beego.AppConfig.St
timezone := beego.AppConfig.St
location, err := time.LoadLoca
if err == nil {
orm.DefaultTimeLoc = l
} else {
beego.Error("加载时区
}
port := beego.AppConfig.String
dataSource := fmt.Sprintf("%s:
if err := orm.RegisterDataBase
beego.Error("注册默认
os.Exit(1)
} else if strings.EqualFold(adapter, "
database := beego.AppConfig.St
if strings.HasPrefix(database,
database = filepath.Jo
if p, err := filepath.Abs(data
database = p
dbPath := filepath.Dir(databas
if _, err := os.Stat(dbPath);
_ = os.MkdirAll(dbPath
err := orm.RegisterDataBase("d
engineercms <--> mindoc
command.go
2020/2/27 23:54:49 15,002 字节 其它一切 UTF-8 UNIX
2019/9/19 17:44:47 14,861 字节 其它一切 UTF-8 UNIX
if err := gocaptcha.ReadFonts(filepath
log.Fatal("读取字体失败 ->", er
// RegisterDataBase()
RegisterCache()
// RegisterModel()
RegisterLogger(conf.LogFile)
ModifyPassword()
//注册缓存管道
func RegisterCache() {
isOpenCache := beego.AppConfig.Default
if !isOpenCache {
cache.Init(&cache.NullCache{})

3.10 update.go



第四步 升级

升级到2.0

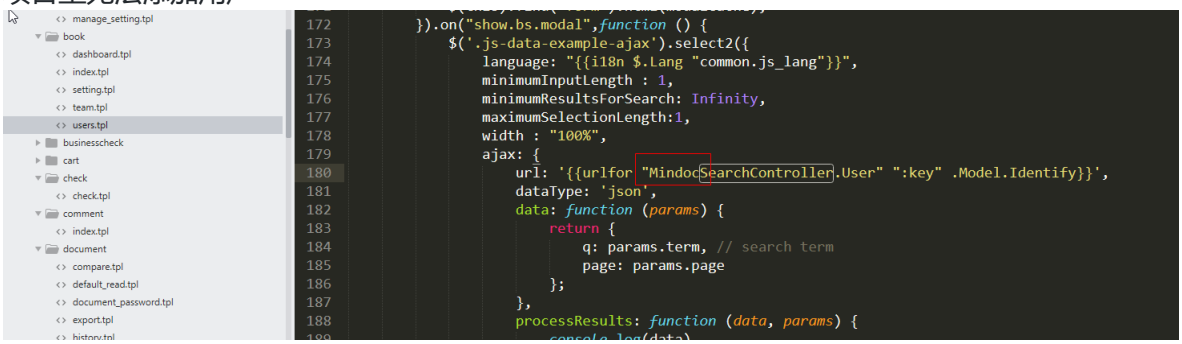
故技重施。
逐项对比修改即可。
mindoc中大量使用urlfor模板函数，它的beego文档在这里：
https://www.bookstack.cn/read/beego-2.0-zh/mvc-view-template.md

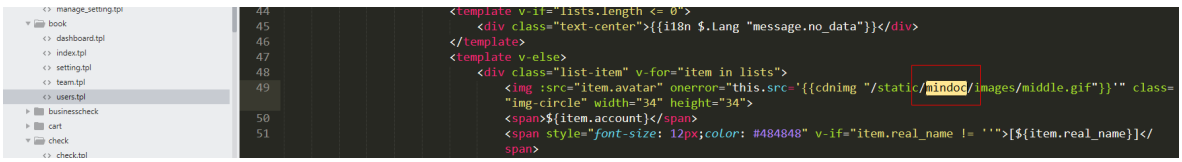
https://www.bookstack.cn/read/beego-2.0-zh/mvc-controller-urlbuilding.md#%E6%A8%A1%E6%9D%BF%E4%B8%AD%E5%A6%82%E4%BD%95%E4%BD%BF%E7%94%A8

项目列表页闪动，缺少默认封面

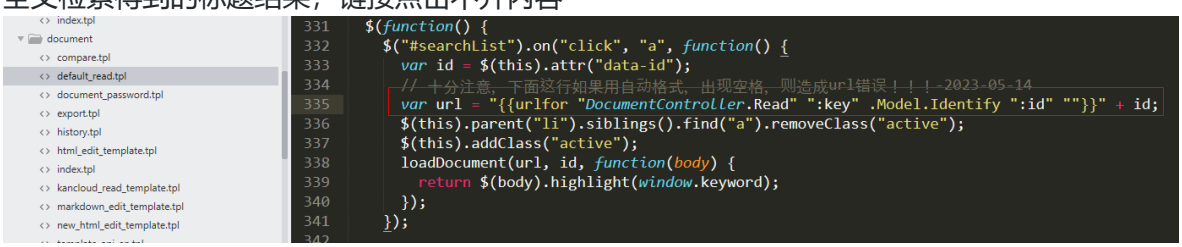


项目里无法添加用户





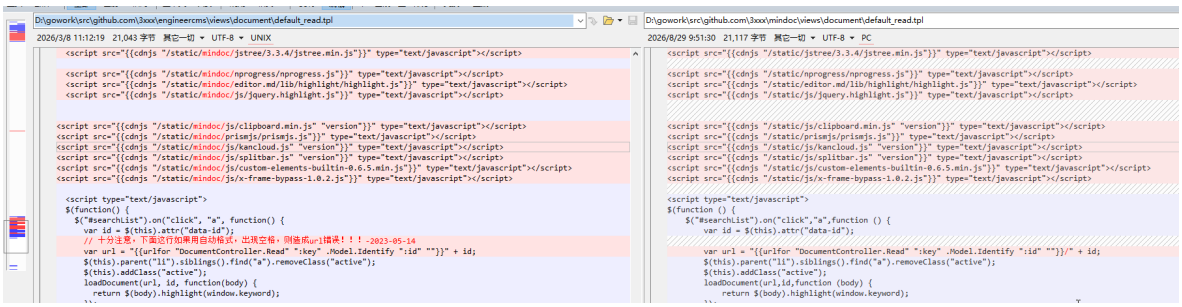
全文检索得到的标题结果，链接点击不开内容



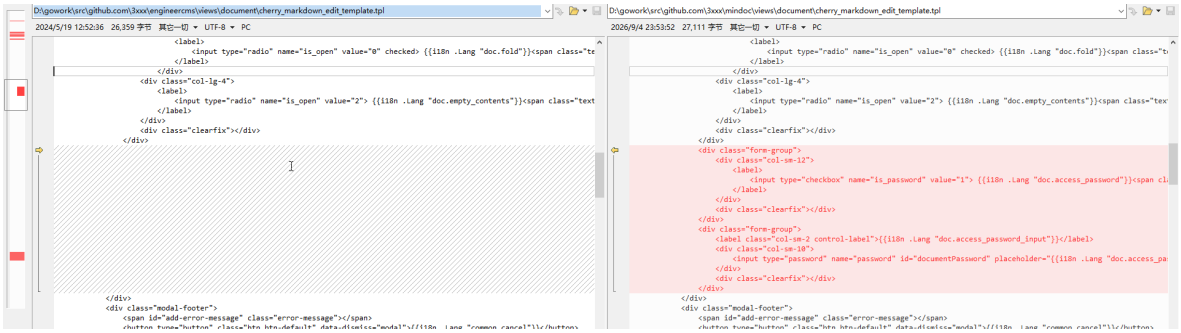
搜索结果无法高亮显示（显示黄底）

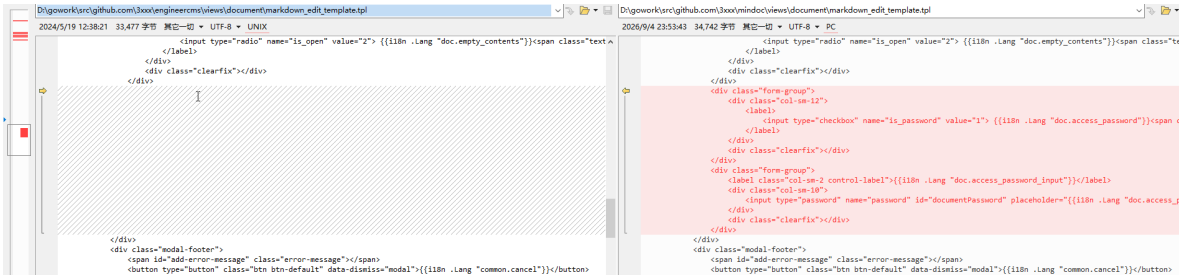
5 20260916记录

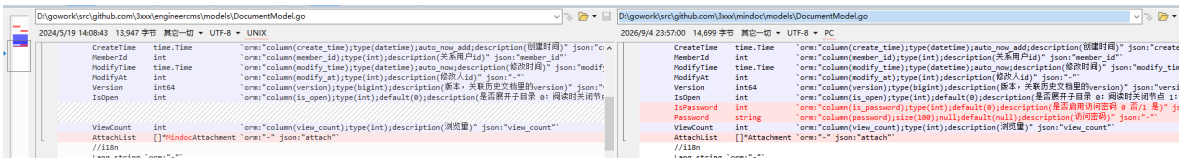
2026年9月16日，将mindoc增加目录密码和增加对“总工笔记deskmindoc”支持而增加了一个book list列表的接口 更新到engineercms里。代码对比如下
有个符号，不知道要不要修改

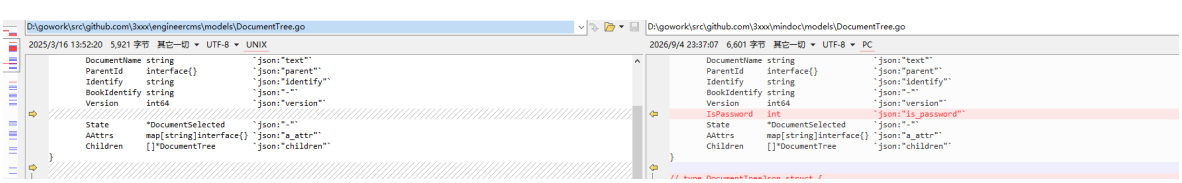


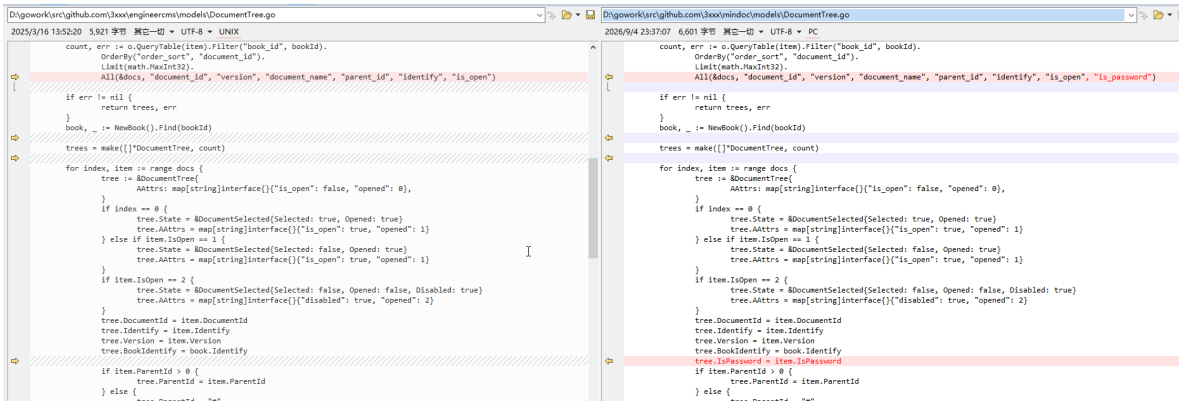
开始进入密码了

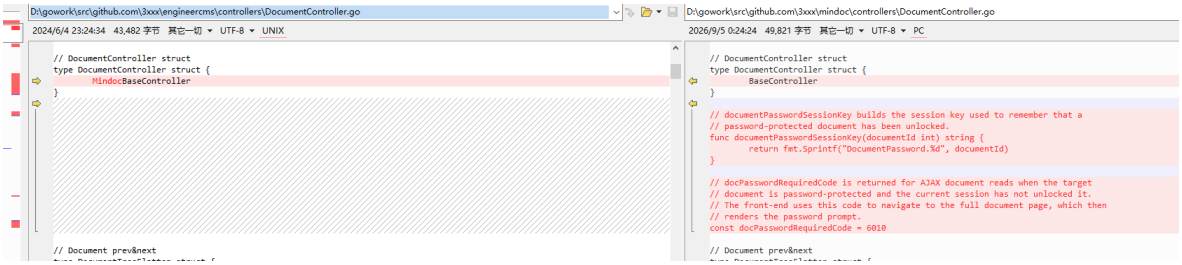


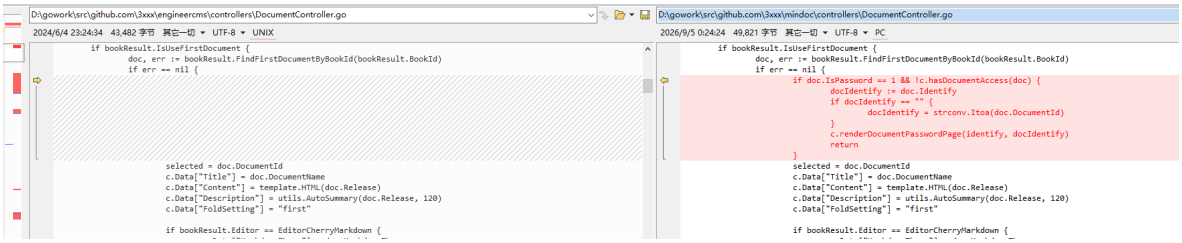


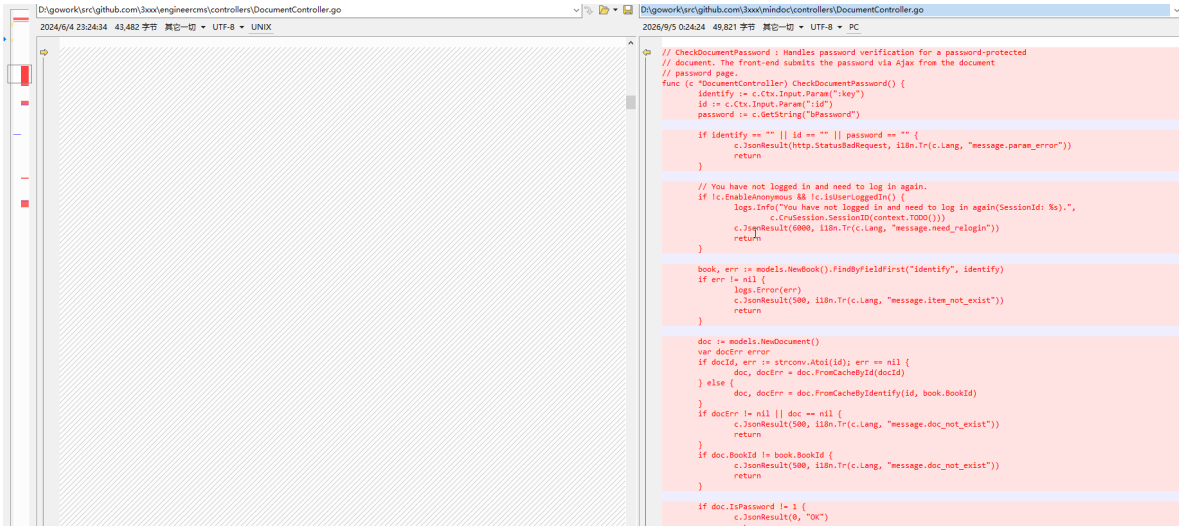


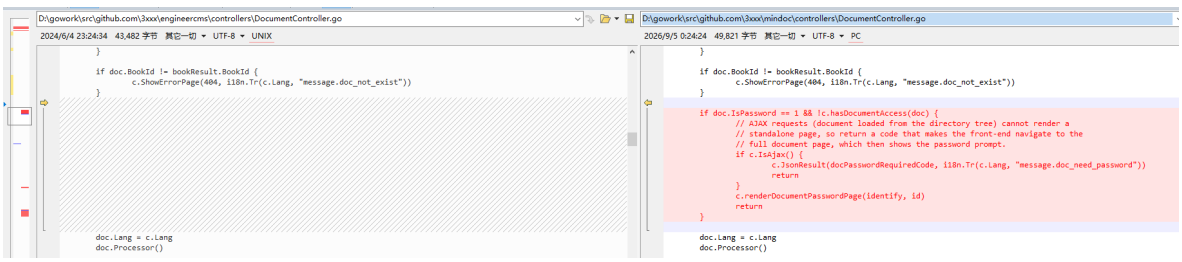


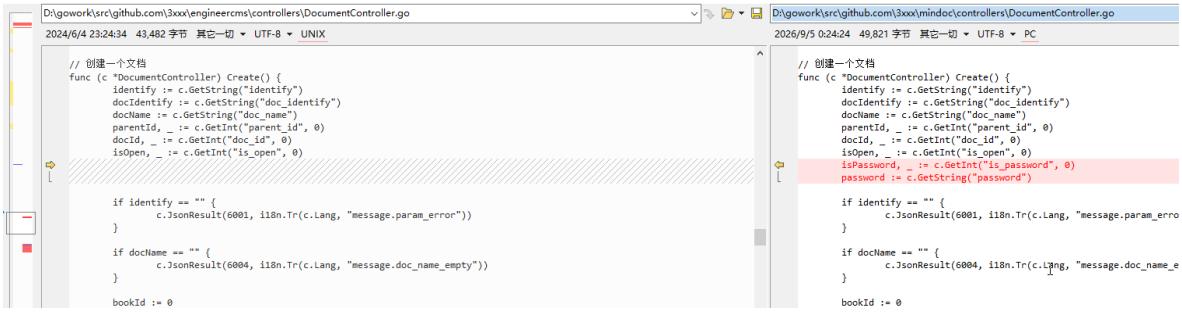


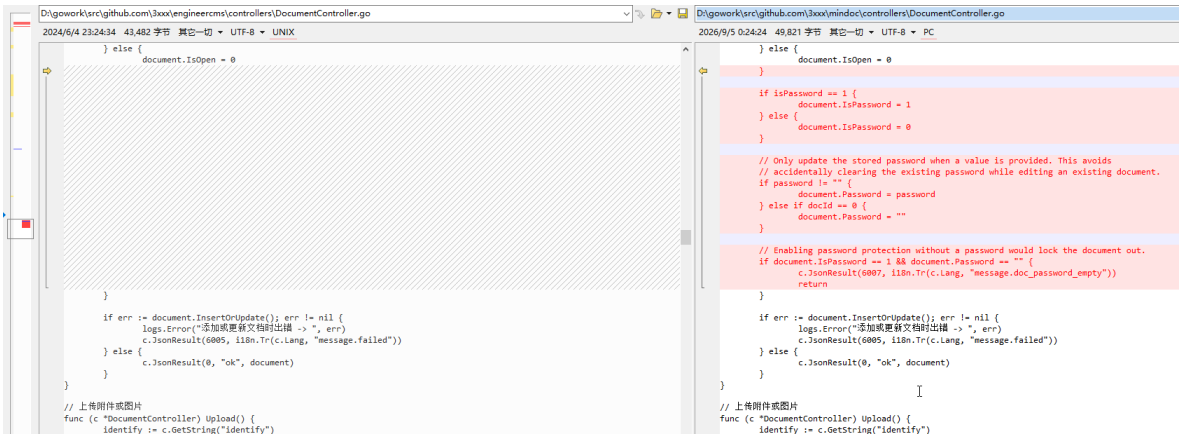


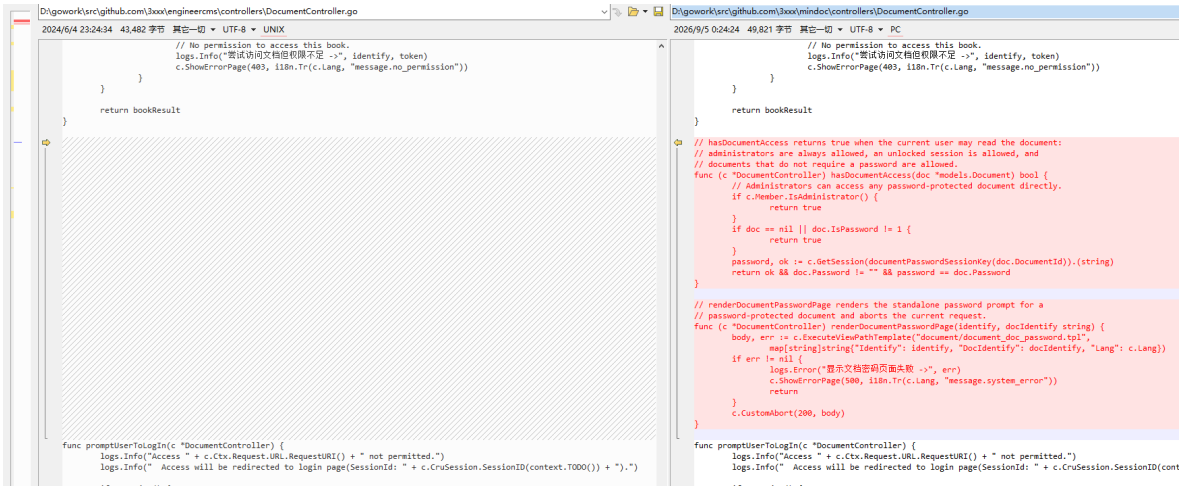


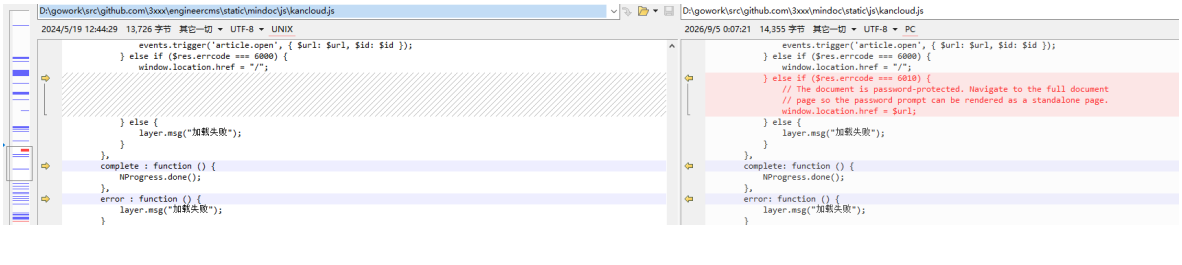


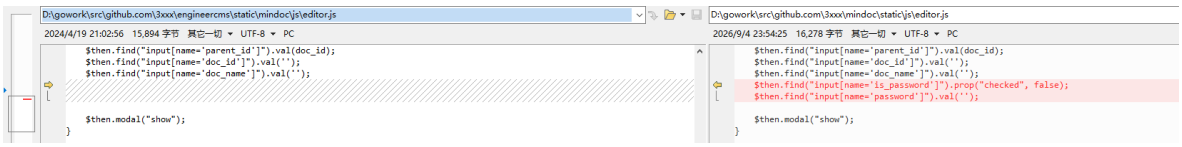


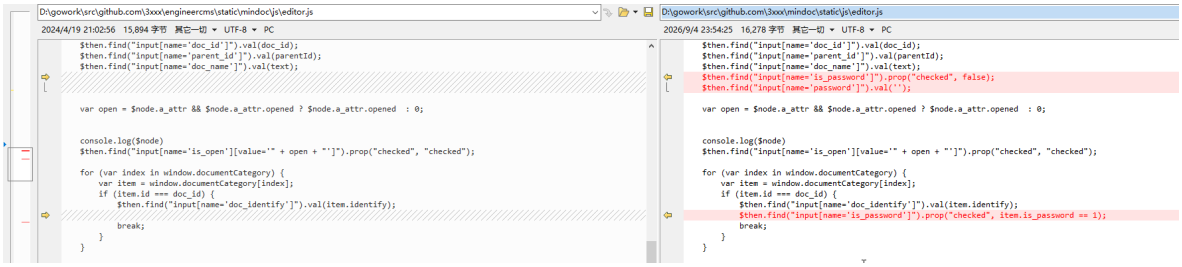

D:\gowork\src\github.com\3xxx\engineercms\controllers\DocumentController.go
2024/6/4 23:24:34 43,482 字节 其它一切 UTF-8 UNIX
D:\gowork\src\github.com\3xxx\mindoc\controllers\DocumentController.go
2026/9/5 0:24:24 49,821 字节 其它一切 UTF-8 PC


D:\gowork\src\github.com\3xxx\engineercms\controllers\DocumentController.go
2024/6/4 23:24:34 43,482 字节 其它一切 UTF-8 UNIX
D:\gowork\src\github.com\3xxx\mindoc\controllers\DocumentController.go
2026/9/5 0:24:24 49,821 字节 其它一切 UTF-8 PC

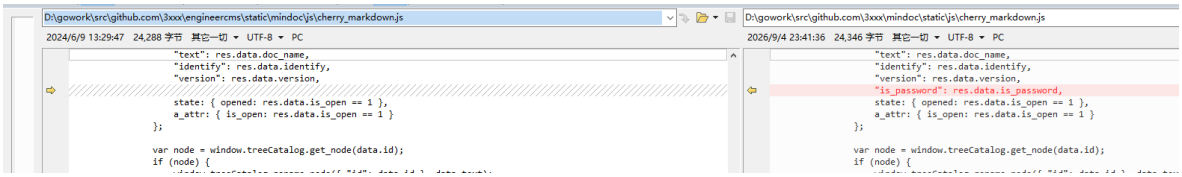

D:\gowork\src\github.com\3xxx\engineercms\controllers\DocumentController.go
2024/6/4 23:24:34 43,482 字节 其它一切 UTF-8 UNIX
D:\gowork\src\github.com\3xxx\mindoc\controllers\DocumentController.go
2026/9/5 0:24:24 49,821 字节 其它一切 UTF-8 PC

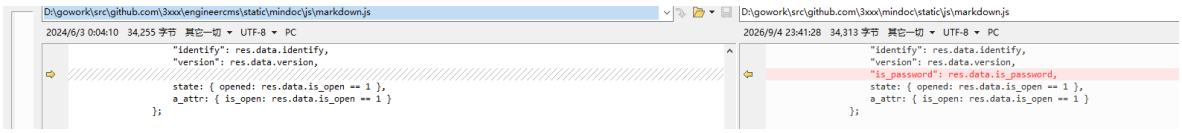

D:\gowork\src\github.com\3xxx\engineercms\static\mindoc\js\kancloud.js
2024/5/19 12:44:29 13,726 字节 其它一切 UTF-8 UNIX
D:\gowork\src\github.com\3xxx\mindoc\static\js\kancloud.js
2026/9/5 0:07:21 14,355 字节 其它一切 UTF-8 PC


D:\gowork\src\github.com\3xxx\engineercms\static\mindoc\js\editor.js
2024/4/19 21:02:56 15,894 字节 其它一切 UTF-8 PC
D:\gowork\src\github.com\3xxx\mindoc\static\js\editor.js
2026/9/4 23:54:25 16,278 字节 其它一切 UTF-8 PC

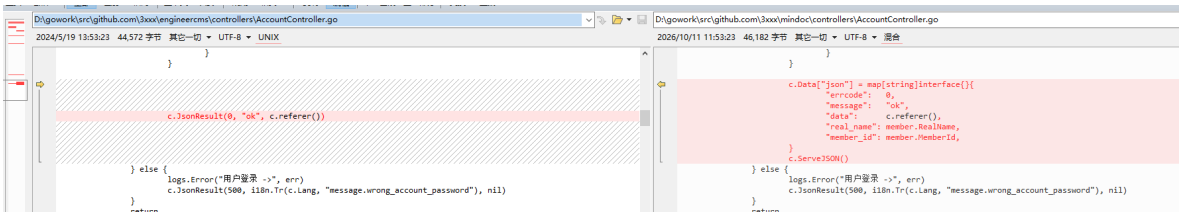




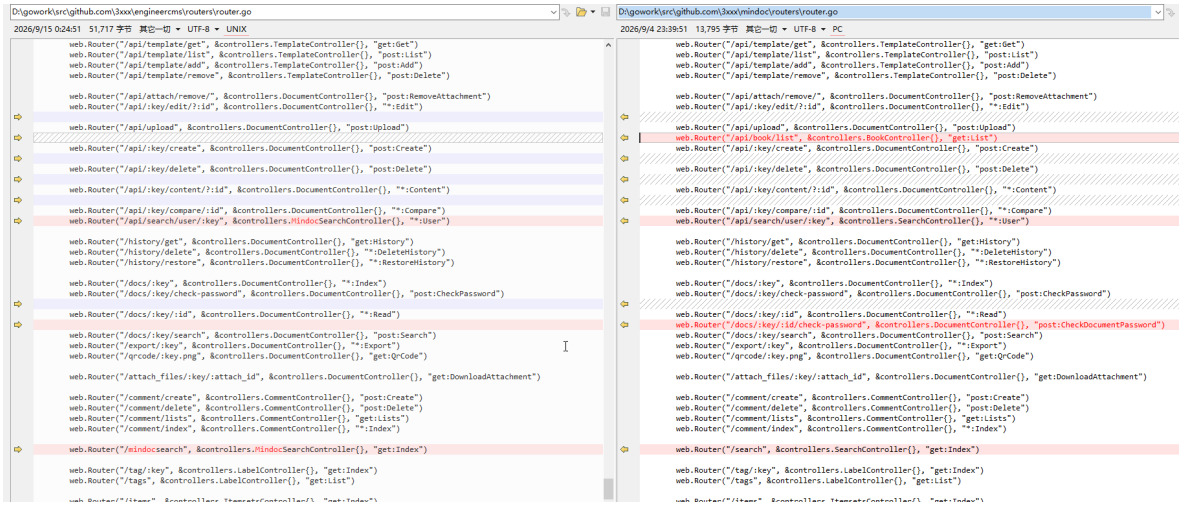


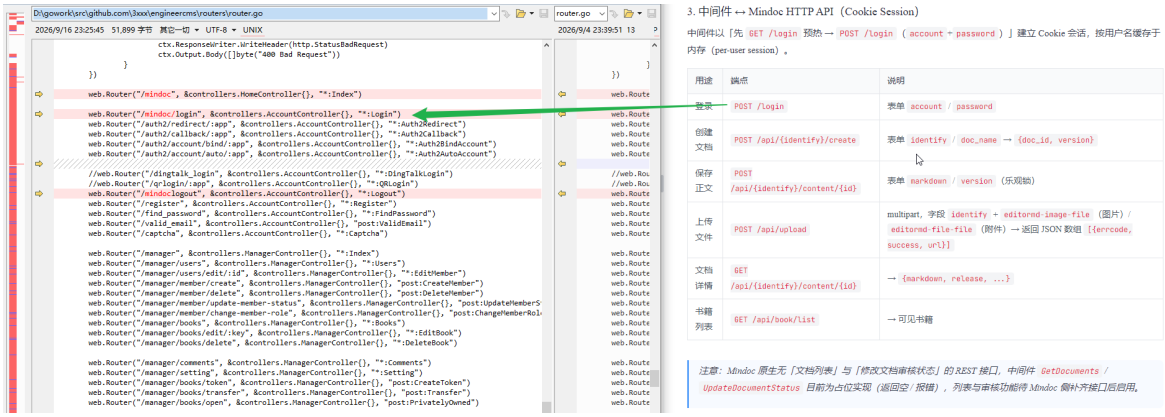


20261011deskmindoc获取登录用户的真实姓名，即昵称

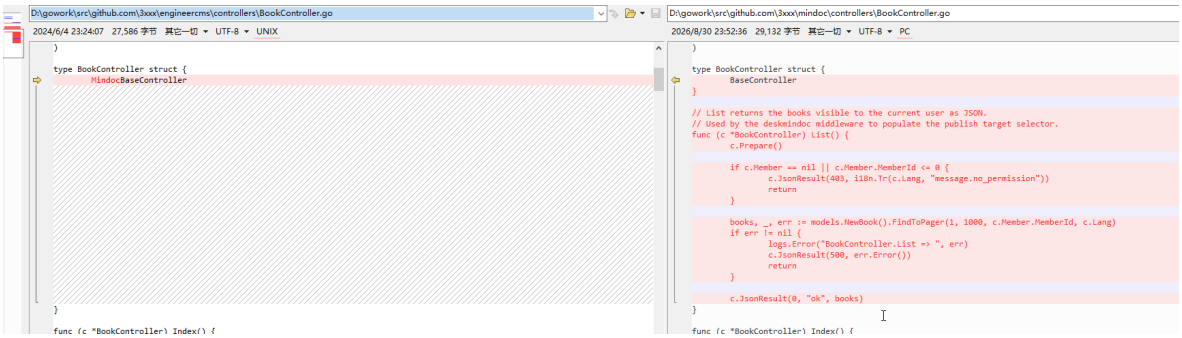


路由：

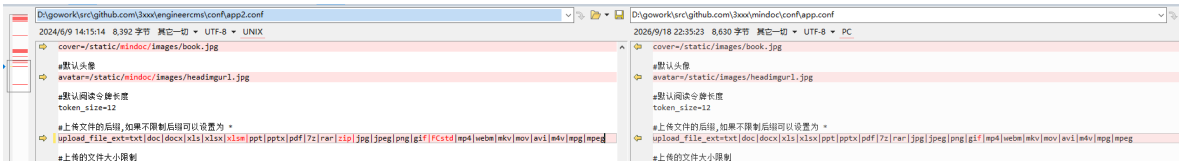


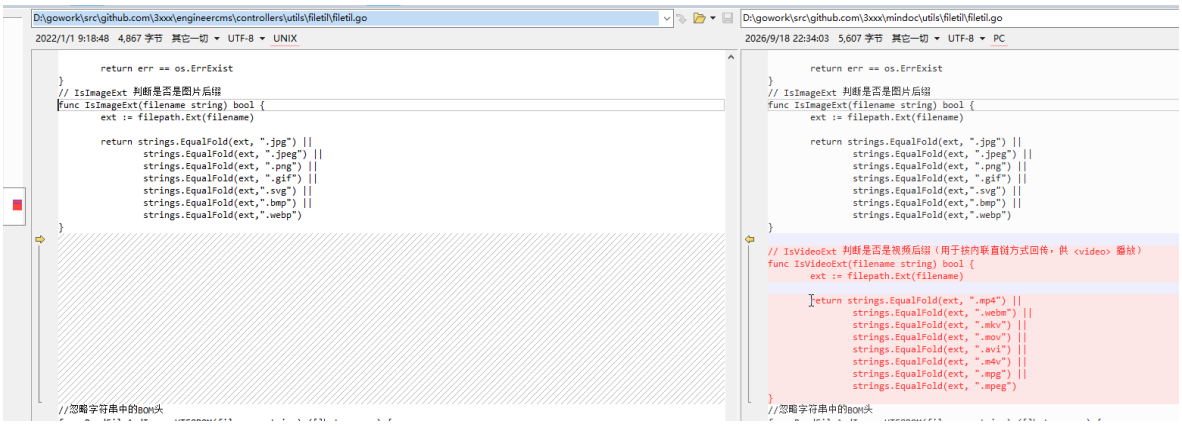


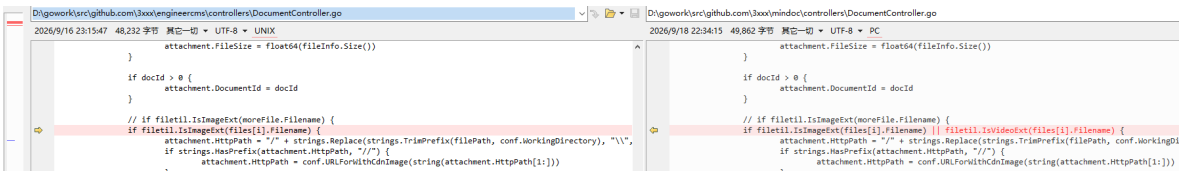
list接口



视频内联播放（改 Mindoc）和附件







5 jstree和ajax请求-添加上一页下一页

在kancloud.js里

```
    initHighlighting();
    window.jsTree = $("#sidebar").jstree({
        'plugins': ["wholerow", "types"],
        "types": {
            "default": {
                "icon": false  // 删除默认图标
            }
        },
        'core': {
            'check_callback': true,
            "multiple": false,
            'animation': 0
        }
    }).on('select_node.jstree', function (node, selected) {
        //如果是空目录则直接触发展开下一级功能
        if (selected.node.a_attr && selected.node.a_attr.disabled) {
            selected.instance.toggle_node(selected.node);
            return false
        }
        $(".m-manual").removeClass('manual-mobile-show-left');
        loadDocument(selected.node.a_attr.href, selected.node.id, selecte
d.node.a_attr['data-version']);
    });
```

如上代码所示，`.on('select_node.jstree', function (node, selected)`是点击目录触发的事件，然后调用`loadDocument`，里面采用ajax请求：

```
function loadDocument($url, $id, $callback) {
    $.ajax({
        url: $url,
        type: "GET",
```

请求到documentcontroller.go里的判断：

```
// 阅读文档
func (c *DocumentController) Read() {
    if c.IsAjax() {
        var data struct {
```

```
    DocId          int     `json:"doc_id"`
    DocIdentify    string  `json:"doc_identify"`
    DocTitle       string  `json:"doc_title"`
    Body           string  `json:"body"`
    Title          string  `json:"title"`
    Version        int64   `json:"version"`
    ViewCount      int     `json:"view_count"`
    MarkdownTheme  string  `json:"markdown_theme"`
    IsMarkdown     bool    `json:"is_markdown"`
```

6 为何只支持一张图片复制粘贴上传

为何明明我们复制了很多图片，包括从word里，从web里，从电脑文件夹内复制了多张图片，可是黏贴到markdown富文本编辑器里，就是只有第一章图片上传了？
问题出在这个剪贴板程序里，剪贴板里是有这些图片存在的，但是这个剪贴板js（clipboard.js）程序只取第一张图片。

```
</!DOCTYPE html>
<html>

<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <style>
  body {
    display: -webkit-flex;
    display: flex;
    -webkit-justify-content: center;
    justify-content: center;
  }

  #tar_box {
    width: 500px;
    height: 500px;
    border: 1px solid red;
  }
  </style>
</head>

<body>
  <textarea></textarea>
  <div contenteditable style="width: 100px;height: 100px; border:1px soli
d">
  </div>
  <script>
  // document.addEventListener('paste', function (event) {
  // console.log(event)
  // })


  document.addEventListener('paste', function(event) {
    console.log(event)
    var isChrome = false;
```

```
    if (event.clipboardData || event.originalEvent) {
      //not for ie11  某些chrome版本使用的是event.originalEvent
      var clipboardData = (event.clipboardData || event.originalEvent.cli
pboardData);
      if (clipboardData.items) {
        // for chrome
        var items = clipboardData.items,
          len = items.length,
          blob = null;
        isChrome = true;
        //items.length比较有意思，初步判断是根据mime类型来的，即有几种mime类
型，长度就是几（待验证）
        //如果粘贴纯文本，那么len=1，如果粘贴网页图片，len=2，items[0].type =
'text/plain'，items[1].type = 'image/*'
        //如果使用截图工具粘贴图片，len=1，items[0].type = 'image/png'
        //如果粘贴纯文本+HTML，len=2，items[0].type = 'text/plain'，items
[1].type = 'text/html'
        // console.log('len:' + len);
        // console.log(items[0]);
        // console.log(items[1]);
        // console.log( 'items[0] kind:', items[0].kind );
        // console.log( 'items[0] MIME type:', items[0].type );
        // console.log( 'items[1] kind:', items[1].kind );
        // console.log( 'items[1] MIME type:', items[1].type );

        //阻止默认行为即不让剪贴板内容在div中显示出来
        event.preventDefault();

        //在items里找粘贴的image,据上面分析,需要循环
        for (var i = 0; i < len; i++) {
          if (items[i].type.indexOf("image") !== -1) {
            // console.log(items[i]);
            // console.log( typeof (items[i]));

            //getAsFile() 此方法只是living standard firefox ie11 并不支持
            blob = items[i].getAsFile();
          }
        }
        if (blob !== null) {
          var reader = new FileReader();
          reader.onload = function(event) {
            // event.target.result 即为图片的Base64编码字符串
            var base64_str = event.target.result
            //可以在这里写上传逻辑 直接将base64编码的字符串上传（可以尝试传入b
lob对象，看看后台程序能否解析）
```

```
            uploadImgFromPaste(base64_str, 'paste', isChrome);
          }
          reader.readAsDataURL(blob);
        }
      } else {
        //for firefox
        setTimeout(function() {
          //设置setTimeout的原因是为了保证图片先插入到div里，然后去获取值
          var imgList = document.querySelectorAll('#tar_box img'),
            len = imgList.length,
            src_str = '',
            i;
          for (i = 0; i < len; i++) {
            if (imgList[i].className !== 'my_img') {
              //如果是截图那么src_str就是base64 如果是复制的其他网页图片那么s
rc_str就是此图片在别人服务器的地址
              src_str = imgList[i].src;
            }
          }
          uploadImgFromPaste(src_str, 'paste', isChrome);
        }, 1);
      }
    } else {
      //for ie11
      setTimeout(function() {
        var imgList = document.querySelectorAll('#tar_box img'),
          len = imgList.length,
          src_str = '',
          i;
        for (i = 0; i < len; i++) {
          if (imgList[i].className !== 'my_img') {
            src_str = imgList[i].src;
          }
        }
        uploadImgFromPaste(src_str, 'paste', isChrome);
      }, 1);
    }
  })

  function uploadImgFromPaste(file, type, isChrome) {
    var formData = new FormData();
    formData.append('image', file);
    formData.append('submission-type', type);

    var xhr = new XMLHttpRequest();
```

```
    xhr.open('POST', '/upload_image_by_paste');
    xhr.onload = function() {
      if (xhr.readyState === 4) {
        if (xhr.status === 200) {
          var data = JSON.parse(xhr.responseText),
            tarBox = document.getElementById('tar_box');
          if (isChrome) {
            var img = document.createElement('img');
            img.className = 'my_img';
            img.src = data.store_path;
            tarBox.appendChild(img);
          } else {
            var imgList = document.querySelectorAll('#tar_box img'),
              len = imgList.length,
              i;
            for (i = 0; i < len; i++) {
              if (imgList[i].className !== 'my_img') {
                imgList[i].className = 'my_img';
                imgList[i].src = data.store_path;
              }
            }
          }

        } else {
          console.log(xhr.statusText);
        }
      };
    };
    xhr.onerror = function(e) {
      console.log(xhr.statusText);
    }
    xhr.send(formData);
  }
  </script>
</body>

</html>
<!-- router.post('/', upload.array(), function (req, res, next) {
//1.获取客户端传来的src_str字符串=>判断是base64还是普通地址=>获取图片类型后缀(jpg/png etc)
//=>如果是base64替换掉"前缀"("data:image\/png; base64," etc)
//2.base64 转为 buffer对象 普通地址则先down下来
//3.写入硬盘(后续可以将地址存入数据库)
//4.返回picture地址
var src_str = req.body.image,
```

```
  timestamp = new Date().getTime();
if ( src_str.match(/^data:image\/png; base64,|^data:image\/jpg; base64,|^da
ta:image\/jpg; base64,|^data:image\/bmp; base64,/) ) {
  //处理截图 src_str为base64字符串
  var pic_suffix = src_str.split(';',1)[0].split('/',2)[1],
    base64 = src_str.replace(/^data:image\/png; base64,|^data:image\/jpg;b
ase64,|^data:image\/jpg; base64,|^data:image\/bmp; base64,/, ''),
    buf = new Buffer(base64, 'base64'),
    store_path = 'public/images/test_' + timestamp + '.' + pic_suffix;

  fs.writeFile(store_path, buf, function (err) {
    if (err) {
      throw err;
    } else {
      res.json({'store_path': store_path});
    }
  });
} else {// 处理非chrome的网页图片 src_str为图片地址
  var temp_array = src_str.split('.'),
    pic_suffix = temp_array[temp_array.length - 1],
    store_path = 'public/images/test_' + timestamp + '.' + pic_suffix,
    wstream = fs.createWriteStream(store_path);

  request(src_str).pipe(wstream);
  wstream.on('finish', function (err) {
    if( err ) {
      throw err;
    } else {
      res.json({"store_path": store_path});
    }
  });
}
}); -->
```

7 批量上传图片

我们选择多张图片首先要改造input支持multiple多选，然后服务端支持批量接收图片，beego框架有解决方案，但是beego的文档中没有写，要去到源码里看。

1 mindoc documentcontroller.go

beego的getfile和getfiles不同，在beego的controller.go中查阅getfile方法，其下方有getfiles的方式，特别注意的是，不能用savetofile方法了，因为savetofile只是对第一个文件进行保存。必须用它代码里提供的例子：io.copy方法。

```
// 上传附件或图片
func (c *DocumentController) Upload() {
    identify := c.GetString("identify")
    docId, _ := c.GetInt("doc_id")
    isAttach := true

    if identify == "" {
        c.JsonResult(6001, i18n.Tr(c.Lang, "message.param_error"))
    }

    name := "editormd-file-file"

    // file, moreFile, err := c.GetFile(name)
    // if err == http.ErrMissingFile || moreFile == nil {
    //     name = "editormd-image-file"
    //     file, moreFile, err = c.GetFile(name)
    //     if err == http.ErrMissingFile || moreFile == nil {
    //         c.JsonResult(6003, i18n.Tr(c.Lang, "message.upload_file_em
pty"))
    //         return
    //     }
    // }
    // ****3xxx
    files, err := c.GetFiles(name)
    if err == http.ErrMissingFile {
        name = "editormd-image-file"
        files, err = c.GetFiles(name)
        if err == http.ErrMissingFile {
            c.JsonResult(6003, i18n.Tr(c.Lang, "message.upload_file_empt
y"))
            return
```

```
        }
    }

    // if err != nil {
    //     http.Error(w, err.Error(), http.StatusNoContent)
    //     return
    // }
    // jMap := make(map[string]interface{})
    // s := []map[int]interface{}{}
    result2 := []map[string]interface{}{}
    for i, _ := range files {
        //for each fileheader, get a handle to the actual file
        file, err := files[i].Open()

        defer file.Close()
        // if err != nil {
        //     http.Error(w, err.Error(), http.StatusInternalServerError)
        //     return
        // }
        //     //create destination file making sure the path is writeabl
e.
        //     dst, err := os.Create("upload/" + files[i].Filename)
        //     defer dst.Close()
        //     if err != nil {
        //         http.Error(w, err.Error(), http.StatusInternalServerEr
ror)
        //         return
        //     }
        //     //copy the uploaded file to the destination file
        //     if _, err := io.Copy(dst, file); err != nil {
        //         http.Error(w, err.Error(), http.StatusInternalServerEr
ror)
        //         return
        //     }
        // }
        // ****

        if err != nil {
            c.JsonResult(6002, err.Error())
        }

        // defer file.Close()

        type Size interface {
            Size() int64
```

```
        }

        // if conf.GetUploadFileSize() > 0 && moreFile.Size > conf.GetUpl
oadFileSize() {
        if conf.GetUploadFileSize() > 0 && files[i].Size > conf.GetUpload
FileSize() {
            c.JsonResult(6009, i18n.Tr(c.Lang, "message.upload_file_size_
limit"))
        }

        // ext := filepath.Ext(moreFile.Filename)
        ext := filepath.Ext(files[i].Filename)
        //文件必须带有后缀名
        if ext == "" {
            c.JsonResult(6003, i18n.Tr(c.Lang, "message.upload_file_type_
error"))
        }
        //如果文件类型设置为 * 标识不限制文件类型
        if conf.IsAllowUploadFileExt(ext) == false {
            c.JsonResult(6004, i18n.Tr(c.Lang, "message.upload_file_type_
error"))
        }

        bookId := 0

        // 如果是超级管理员，则不判断权限
        if c.Member.IsAdministrator() {
            book, err := models.NewBook().FindByFieldFirst("identify", id
entify)

            if err != nil {
                c.JsonResult(6006, i18n.Tr(c.Lang, "message.doc_not_exist
_or_no_permit"))
            }

            bookId = book.BookId
        } else {
            book, err := models.NewBookResult().FindByIdentify(identify,
c.Member.MemberId)

            if err != nil {
                logs.Error("DocumentController.Edit => ", err)
                if err == orm.ErrNoRows {
                    c.JsonResult(6006, i18n.Tr(c.Lang, "message.no_permis
sion"))
```

```
                }

                c.JsonResult(6001, err.Error())
            }

            // 如果没有编辑权限
            if book.RoleId != conf.BookEditor && book.RoleId != conf.Book
Admin && book.RoleId != conf.BookFounder {
                c.JsonResult(6006, i18n.Tr(c.Lang, "message.no_permissio
n"))
            }

            bookId = book.BookId
        }

        if docId > 0 {
            doc, err := models.NewDocument().Find(docId)
            if err != nil {
                c.JsonResult(6007, i18n.Tr(c.Lang, "message.doc_not_exis
t"))
            }

            if doc.BookId != bookId {
                c.JsonResult(6008, i18n.Tr(c.Lang, "message.doc_not_belon
g_project"))
            }
        }

        fileName := "m_" + cryptil.UniqueId() + "_r"
        filePath := filepath.Join(conf.WorkingDirectory, "uploads", ident
ify)

        //将图片和文件分开存放
        // if filetil.IsImageExt(moreFile.Filename) {
        if filetil.IsImageExt(files[i].Filename) {
            filePath = filepath.Join(filePath, "images", fileName+ext)
        } else {
            filePath = filepath.Join(filePath, "files", fileName+ext)
        }

        path := filepath.Dir(filePath)

        _ = os.MkdirAll(path, os.ModePerm)

        // err = c.SaveToFile(name, filePath) // frome beego controller.g
```

```
o: savetofile it only operates the first one of mutil-upload form file fi
eld.

        //copy the uploaded file to the destination file
        dst, err := os.Create(filePath)
        defer dst.Close()
        if _, err := io.Copy(dst, file); err != nil {
            logs.Error("保存文件失败 -> ", err)
            c.JsonResult(6005, i18n.Tr(c.Lang, "message.failed"))
        }

        // if err != nil {
        //     logs.Error("保存文件失败 -> ", err)
        //     c.JsonResult(6005, i18n.Tr(c.Lang, "message.failed"))
        // }

        attachment := models.NewAttachment()
        attachment.BookId = bookId
        // attachment.FileName = moreFile.Filename
        attachment.FileName = files[i].Filename
        attachment.CreateAt = c.Member.MemberId
        attachment.FileExt = ext
        attachment.FilePath = strings.TrimPrefix(filePath, conf.WorkingDi
rectory)
        attachment.DocumentId = docId

        if fileInfo, err := os.Stat(filePath); err == nil {
            attachment.FileSize = float64(fileInfo.Size())
        }

        if docId > 0 {
            attachment.DocumentId = docId
        }

        // if filetil.IsImageExt(moreFile.Filename) {
        if filetil.IsImageExt(files[i].Filename) {
            attachment.HttpPath = "/" + strings.Replace(strings.TrimPrefi
x(filePath, conf.WorkingDirectory), "\\", "/", -1)
            if strings.HasPrefix(attachment.HttpPath, "//") {
                attachment.HttpPath = conf.URLForWithCdnImage(string(atta
chment.HttpPath[1:]))
            }

            isAttach = false
        }
```

```
        err = attachment.Insert()

        if err != nil {
            os.Remove(filePath)
            logs.Error("文件保存失败 ->", err)
            c.JsonResult(6006, i18n.Tr(c.Lang, "message.failed"))
        }

        if attachment.HttpPath == "" {
            attachment.HttpPath = conf.URLForNotHost("DocumentController.
DownloadAttachment", ":key", identify, ":attach_id", attachment.Attachmen
tId)

            if err := attachment.Update(); err != nil {
                logs.Error("保存文件失败 ->", err)
                c.JsonResult(6005, i18n.Tr(c.Lang, "message.failed"))
            }
        }
        result := map[string]interface{}{
            "errcode":   0,
            "success":   1,
            "message":   "ok",
            "url":       attachment.HttpPath,
            "alt":       attachment.FileName,
            "is_attach": isAttach,
            "attach":    attachment,
        }
        result2 = append(result2, result)
    }

    c.Ctx.Output.JSON(result2, true, false)
    c.StopRun()
}
```

image-dialog.js

位置在mindoc\static\editor.md\plugins\image-dialog\image-dialog.js

```
/*!
 * Image (upload) dialog plugin for Editor.md
 *
 * @file        image-dialog.js
```

```
 * @author        pandao
 * @version       1.3.4
 * @updateTime    2015-06-09
 * {@link        https://github.com/pandao/editor.md}
 * @license       MIT
 */
(function() {
  var factory = function(exports) {
    var pluginName = "image-dialog";
    exports.fn.imageDialog = function() {
      var _this = this;
      var cm = this.cm;
      var lang = this.lang;
      var editor = this.editor;
      var settings = this.settings;
      var cursor = cm.getCursor();
      var selection = cm.getSelection();
      var imageLang = lang.dialog.image;
      var classPrefix = this.classPrefix;
      var iframeName = classPrefix + "image-iframe";
      var dialogName = classPrefix + pluginName,
        dialog;

      cm.focus();

      var loading = function(show) {
        var _loading = dialog.find("." + classPrefix + "dialog-mask");
        _loading[(show) ? "show" : "hide"]();
      };

      if (editor.find("." + dialogName).length < 1) {
        var guid = (new Date).getTime();
        var action = settings.imageUploadURL + (settings.imageUploadURL.i
ndexOf("?") >= 0 ? "&" : "?") + "guid=" + guid;

        if (settings.crossDomainUpload) {
          action += "&callback=" + settings.uploadCallbackURL + "&dialog_
id=editormd-image-dialog-" + guid;
        }

        var dialogContent = ((settings.imageUpload) ? "<form action=\"" +
action + "\" target=\"" + iframeName + "\" method=\"post\" enctype=\"mult
ipart/form-data\" class=\"" + classPrefix + "form\">" : "<div class=\"" +
classPrefix + "form\">") +
          ((settings.imageUpload) ? "<iframe name=\"" + iframeName + "\"
```

```
id=\"" + iframeName + "\" guid=\"" + guid + "\"></iframe>" : "") +
          "<label>" + imageLang.url + "</label>" +
          "<input type=\"text\" data-url />" + (function() {
            return (settings.imageUpload) ? "<div class=\"" + classPrefix
+ "file-input\">" +
              // 3xxx 下行添加multiple=\"multiple\"
              "<input type=\"file\" name=\"" + classPrefix + "image-file
\" accept=\"image/jpeg,image/png,image/gif,image/jpg\" multiple=\"multipl
e\" />" +
              "<input type=\"submit\" value=\"" + imageLang.uploadButton
+ "\" />" +
              "</div>" : "";
          })() +
          "<br/>" +
          "<label>" + imageLang.alt + "</label>" +
          "<input type=\"text\" value=\"" + selection + "\" data-alt />"
+
          "<br/>" +
          "<label>" + imageLang.link + "</label>" +
          "<input type=\"text\" value=\"http://\" data-link />" +
          "<br/>" +
          ((settings.imageUpload) ? "</form>" : "</div>");
        //var imageFooterHTML = "<button class=\"" + classPrefix + "btn "
+ classPrefix + "image-manager-btn\" style=\"float:left;\">" + imageLang.
managerButton + "</button>";
        dialog = this.createDialog({
          title: imageLang.title,
          width: (settings.imageUpload) ? 465 : 380,
          height: 254,
          name: dialogName,
          content: dialogContent,
          mask: settings.dialogShowMask,
          drag: settings.dialogDraggable,
          lockScreen: settings.dialogLockScreen,
          maskStyle: {
            opacity: settings.dialogMaskOpacity,
            backgroundColor: settings.dialogMaskBgColor
          },
          // 这里将多图片地址改造后插入文档中
          buttons: {
            enter: [lang.buttons.enter, function() {
              var url = this.find("[data-url]").val();
              var alt = this.find("[data-alt]").val();
              var link = this.find("[data-link]").val();
              if (url === "") {
```

```
                alert(imageLang.imageURLEmpty);
                return false;
              }
              // 这里增加循环
              let arr = url.split(";");
              var altAttr = (alt !== "") ? " \"" + alt + "\"" : "";
              for (let i = 0; i < arr.length; i++) {
                if (link === "" || link === "http://") {
                  // cm.replaceSelection("![" + alt + "](" + url + altAtt
r + ")");
                  cm.replaceSelection("![" + alt + "](" + arr[i] + altAtt
r + ")");
                } else {
                  // cm.replaceSelection("[![" + alt + "](" + url + altAt
tr + ")](" + link + altAttr + ")");
                  cm.replaceSelection("[![" + alt + "](" + arr[i] + altAt
tr + ")](" + link + altAttr + ")");
                }
              }
              if (alt === "") {
                cm.setCursor(cursor.line, cursor.ch + 2);
              }
              this.hide().lockScreen(false).hideMask();
              return false;
            }],

            cancel: [lang.buttons.cancel, function() {
              this.hide().lockScreen(false).hideMask();
              return false;
            }]
          }
        });
        dialog.attr("id", classPrefix + "image-dialog-" + guid);
        if (!settings.imageUpload) {
          return;
        }
        var fileInput = dialog.find("[name=\"" + classPrefix + "image-fil
e\"]");
        fileInput.bind("change", function() {
          // 3xxx 20240602
          // let formData = new FormData();
          // 获取文本框dom
          // var doc = document.getElementById('doc');
          // 获取上传控件dom
          // var upload = document.getElementById('upload');
```

```
            // let files = upload.files;
            //遍历文件信息append到formData存储
            // for (let i = 0; i < files.length; i++) {
            //     let file = files[i]
            //     formData.append('files', file)
            // }
            // 获取文件名
            // var fileName = upload.files[0].name;
            // 获取文件路径
            // var filePath = upload.value;
            // doc.value = fileName;
            // 3xxx
            console.log(fileInput);
            console.log(fileInput[0].files);
            let files = fileInput[0].files;
            for (let i = 0; i < files.length; i++) {
              var fileName = files[i].name;
              // var fileName  = fileInput.val();
              var isImage = new RegExp("(\\.(" + settings.imageFormats.join
("|") + "))$"); // /(\.(webp|jpg|jpeg|gif|bmp|png))$/
              if (fileName === "") {
                alert(imageLang.uploadFileEmpty);
                return false;
              }
              if (!isImage.test(fileName)) {
                alert(imageLang.formatNotAllowed + settings.imageFormats.jo
in(", "));
                return false;
              }
              loading(true);
              var submitHandler = function() {
                var uploadIframe = document.getElementById(iframeName);
                uploadIframe.onload = function() {
                  loading(false);
                  var body = (uploadIframe.contentWindow ? uploadIframe.con
tentWindow : uploadIframe.contentDocument).document.body;
                  var json = (body.innerText) ? body.innerText : ((body.tex
tContent) ? body.textContent : null);
                  json = (typeof JSON.parse !== "undefined") ? JSON.parse(j
son) : eval("(" + json + ")");
                  var url="";
                  for (let i = 0; i < json.length; i++) {
                    if (json[i].success === 1) {
                      if (i==0){
                        url=json[i].url;
```

```
                    }else{
                      url=url+";"+json[i].url;
                    }
                  } else {
                    alert(json[i].message);
                  }
                }
                dialog.find("[data-url]").val(url)
                return false;
              };
            };
            dialog.find("[type=\"submit\"]").bind("click", submitHandle
r).trigger("click");
          }
        });
      }
      dialog = editor.find("." + dialogName);
      dialog.find("[type=\"text\"]").val("");
      dialog.find("[type=\"file\"]").val("");
      dialog.find("[data-link]").val("http://");
      this.dialogShowMask(dialog);
      this.dialogLockScreen();
      dialog.show();
    };
  };

  // CommonJS/Node.js
  if (typeof require === "function" && typeof exports === "object" && typ
eof module === "object") {
    module.exports = factory;
  } else if (typeof define === "function") // AMD/CMD/Sea.js
  {
    if (define.amd) { // for Require.js
      define(["editormd"], function(editormd) {
        factory(editormd);
      });
    } else { // for Sea.js
      define(function(require) {
        var editormd = require("./../../editormd");
        factory(editormd);
      });
    }
  } else {
    factory(window.editormd);
```

```
  }
})();
```

func (c *DocumentController) Upload() 接口返回调整成数组后，影响单图复制粘贴功能

修改以下
mindoc\static\js\blog.js
mindoc\static\js\markdown.js
mindoc\static\js\quill.js

```
uploadImage("docEditor", function ($state, $res) {
    if ($state === "before") {
        return layer.load(1, {
            shade: [0.1, '#fff'] // 0.1 透明度的白色背景
        });
    } else if ($state === "success") {
        // if ($res.errcode === 0) {
            // var value = '![](' + $res.url + ')';
        // 3xxx 20240602
        if ($res[0].errcode === 0) {
            var value = '![](' + $res[0].url + ')';
            window.editor.insertValue(value);
        }
    }
});

uploadImage("docEditor", function ($state, $res) {
    if ($state === "before") {
        return layer.load(1, {
            shade: [0.1, '#fff'] // 0.1 透明度的白色背景
        });
    } else if ($state === "success") {
        // if ($res.errcode === 0) {
        if ($res[0].errcode === 0) {
            var range = window.editor.getSelection();
            // window.editor.insertEmbed(range.index, 'image', $res.url);
            window.editor.insertEmbed(range.index, 'image', $res[0].url);
        }
    }
});
```

cherry

cherry的批量上传图片与上述方法不同，上面是批量图片一次性由服务端上传，然后返回数组，再对数组进行循环处理，插入编辑器.....这个cherry是将多图拆开循环，然后利用cherry_markdown里的myFileUpload 一个个上传。
能否参考上述一次性上传多图，然后服务端返回数组，再处理数组：循环和修改?
cherry-markdown.js里的handleUpload方法里构造input输入框来多选图片，点击后来到var Image$2里的_createClass里调用handleUpload方法，然后到handleUpload里获得选择的文件，记得更新markdown.css文件
cherry-markdown.js

```
    /**
     * 上传文件的逻辑，如果有callback，则不再走默认的替换文本的逻辑，而是调用callback
     * @param {string} type 上传文件的类型
     */
    function handleUpload(editor) {
      var type = arguments.length > 1 && arguments[1] !== undefined ? arguments[1] : 'image';
      var accept = arguments.length > 2 && arguments[2] !== undefined ? arguments[2] : '*';
      // type为上传文件类型 image|video|audio|pdf|word
      var input = document.createElement('input');
      input.type = 'file';
      input.id = 'fileUpload';
      input.value = '';
      input.style.display = 'none';
      input.accept = accept; // document.body.appendChild(input);
      input.multiple = 'multiple';

    input.addEventListener('change', function (event) {
        // @ts-ignore
        // var _event$target$files = _slicedToArray(event.target.files, 1),
        // file = _event$target$files[0]; // 文件上传后的回调函数可以由调用方自己实现
        // 3xxx 20240607 这里增加对多个文件的循环
           let files = event.target.files;
            for (let i = 0; i < files.length; i++) {

              var file = files[i]
              editor.options.fileUpload(file, function(url) { // 这里会进入cherry_markdown.js
```

```
                var params = arguments.length > 1 && arguments[1] !== und
efined ? arguments[1] : {};
```

`editor.options.fileUpload(file, function(url) {` 设断点，F11进入，会来到cherry_markdown.js最后位置 `myFileUpload`

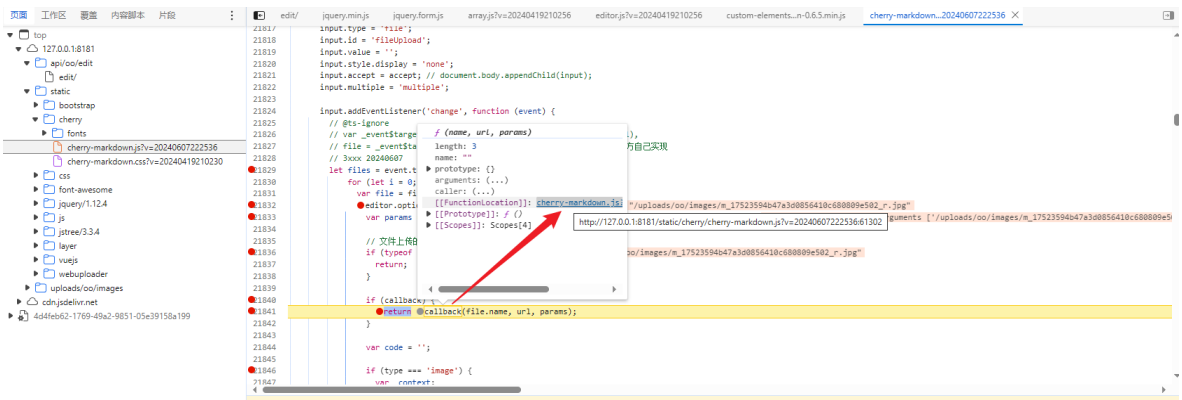
```
function myFileUpload(file, callback) {
    // 创建 FormData 对象以便包含要上传的文件
    var formData = new FormData();
    formData.append("editormd-file-file", file); // "file" 是与你的服务端接口相对应的字段名
    var layerIndex = 0;
    // AJAX 请求
    $.ajax({
        url: window.fileUploadURL, // 确保此 URL 是文件上传 API 的正确 URL
        type: "POST",
        async: false, // 3xxx 20240609这里修改为同步，保证cherry批量上传图片时，插入的图片名称是正确的，否则，插入的图片名称都是最后一个名称
        dataType: "json",
        data: formData,
        processData: false, // 必须设置为 false，因为数据是 FormData 对象，不需要对数据进行序列化处理
        contentType: false, // 必须设置为 false，因为是 FormData 对象，jQuery 将不会设置内容类型头
        beforeSend: function () {
            layerIndex = layer.load(1, {
                shade: [0.1, '#fff'] // 0.1 透明度的白色背景
            });
        },
        error: function () {
            layer.close(layerIndex);
            layer.msg(locales[lang].uploadFailed);
        },
        success: function (data) {
            layer.close(layerIndex);
            if (data[0].errcode !== 0) { // 3xxx 这几个data都要改成data[0]
                layer.msg(data[0].message);
            } else {
                callback(data[0].url); // 假设返回的 JSON 中包含上传文件的 URL，调用回调函数并传入 URL
            }
        }
    });
```

循环赋值给params，再通过callback进入回到var Image$2方法里的 ——_createClass里，不断修改var finalName = params.name ? params.name : name;

插入图片的菜单在这里cherry_markdown.js

```
'|',
                'formula',
                {
                    insert: ['image', 'audio', 'video', 'link', 'hr', 'br', 'code', 'formula', 'toc', 'table', 'pdf', 'word', 'ruby'],
                },
                'graph',
```

这个菜单按钮的动作会来到cherry-markdown.js





cherry-markdown.js约61296行会随着上传的图片，不断修改编辑器中的图片链接地址

```
/**
     * 插入图片
     */
    var Image$2 = /*#__PURE__*/function (_MenuBase) {
      _inherits(Image, _MenuBase);
```

```
      var _super = _createSuper$11(Image);
      function Image($cherry) {
        var _this;
        _classCallCheck(this, Image);
        _this = _super.call(this, $cherry);
        _this.setName('image', 'image');
        return _this;
      }
      /**
       * 响应点击事件
       * @param {string} selection 被用户选中的文本内容
       * @returns {string} 回填到编辑器光标位置/选中文本区域的内容
       */

      _createClass(Image, [{
        key: "onClick",
        value: function onClick(selection) {
          var _this2 = this,
              _this$$cherry$options,
              _this$$cherry$options2,
              _this$$cherry$options3;
          if (this.hasCacheOnce()) {
            var _context, _context2;
            // @ts-ignore
            var _this$getAndCleanCach = this.getAndCleanCacheOnce(),
                name = _this$getAndCleanCach.name,
                url = _this$getAndCleanCach.url,
                params = _this$getAndCleanCach.params;
            var begin = '![';
            var end = "](".concat(url, ")");
            this.registerAfterClickCb(function () {
              _this2.setLessSelection(begin, end);
            });
            var finalName = params.name ? params.name : name;
            return concat$5(_context = concat$5(_context2 = "".concat(beg
in).concat(finalName)).call(_context2, handelParams(params))).call(_conte
xt, end);
          }
          var accept = (_this$$cherry$options = (_this$$cherry$options2 =
this.$cherry.options) === null || _this$$cherry$options2 === void 0 ? voi
d 0 : (_this$$cherry$options3 = _this$$cherry$options2.fileTypeLimitMap)
=== null || _this$$cherry$options3 === void 0 ? void 0 : _this$$cherry$op
tions3.image) !== null && _this$$cherry$options !== void 0 ? _this$$cherr
y$options : '*'; // 插入图片，调用上传文件逻辑
```

```
          handleUpload(this.editor, 'image', accept, function (name, url,
params) {
            _this2.setCacheOnce({
              name: name,
              url: url,
              params: params
            });
            _this2.fire(null);
          });
          this.updateMarkdown = false;
          return selection;
        }
```

8 添加froala富文本编辑器

目的和用法

1. 复制word图文到mindoc的froala编辑器，批量图片自动转存到mindoc服务器。
2. 复制web图文到mindoc的froala编辑器，批量图片自动转存到mindoc服务器。
3. 单纯的图片批量上传froala编辑器不支持（无论是按钮上传还是多图片复制黏贴上传都不支持），所以还是切换到markdown编辑器进行图片批量上传（按钮上传）。

互联网的记忆只有10年，超过10年，随着技术的进步和人员的离去，很多知识会消失，例如bbs已经消失，那些年bbs上的资料也随着服务器没有维护而消失；还有很多个人博客也会随着技术进步换成了另外的方式，个人服务器到期而消失；例如qq zone即qq空间，由于其封闭性，采用flash开发，技术老旧，无法检索到，知识封闭，也消失了。所以，对于好的知识，应该鼓励尽量多人去复制保存下来，多平台展示。

4. 使用方法：首先在项目设置里选择frolala编辑器，保存。然后进人项目编辑就可以利用froala编辑器的word和web图文转存功能了。
5. 存在的问题：
   5.1 该编辑器没有调得完美，所以从功能角度，仅用于对付批量图文转存，具体编辑我还是喜欢markdown的编辑器。
   有时候word图文无法转存，是因为word可能不是docx格式，doc格式另存为docx好像也不行，好像必须是一开始就是docx格式。
   有时候web图文无法转存，例如bing里的新闻页面，图文转存后会出现蒙皮，百度的一些新闻页面里的png/jpg文件名后跟着一串字符，无法转存。
   5.2 github.com/mindoc上的代码不带froala的css和js文件，需要用户自行下载放到mindoc的static/froala里。

第一天

跟踪流程——试想，有没有一种工具能够将下面这个流程直接给出来呢？要花挺多时间来人工读代码。

项目设置

首先是项目设置（book/setting.tpl）里，选择某个编辑器

```
<form method="post" id="bookEditForm" action="{{urlfor "BookController.SaveBook"}}">
```

```
                <div class="form-group">
                  <label>{{i18n $.Lang "blog.text_editor"}}</label>
                  <div class="radio">
                    <label class="radio-inline">
                      <input type="radio" {{if eq .Model.Editor "markdow
n"}} checked{{end}} name="editor" value="markdown"> Markdown {{i18n $.Lan
g "blog.text_editor"}}
                    </label>
                    <label class="radio-inline">
                      <input type="radio" {{if eq .Model.Editor "cherry_m
arkdown"}} checked{{end}} name="editor" value="cherry_markdown"> Markdown
{{i18n $.Lang "blog.text_editor"}}(cherry)
                    </label>
                    <label class="radio-inline">
                      <input type="radio" {{if eq .Model.Editor "new_htm
l"}} checked{{end}} name="editor" value="new_html"> Html {{i18n $.Lang "b
log.text_editor"}}(Quill)
                    </label>
                    <label class="radio-inline">
                      <input type="radio" {{if eq .Model.Editor "html"}}
checked{{end}} name="editor" value="html"> Html {{i18n $.Lang "blog.text_
editor"}}(wangEditor)
                    </label>
                    <!-- 3xxx 20240603 -->
                    <label class="radio-inline">
                      <input type="radio" {{if eq .Model.Editor "froal
a"}} checked{{end}} name="editor" value="froala"> Froala {{i18n $.Lang "b
log.text_editor"}}
                    </label>
                  </div>
                </div>
```

这里选择了之后，确定保存会传给 `BookController.SaveBook`

```
// 保存项目信息
func (c *BookController) SaveBook() {

    editor := strings.TrimSpace(c.GetString("editor"))

    if err := book.Update(); err != nil {
        c.JsonResult(6006, i18n.Tr(c.Lang, "message.failed"))
    }
```

保存到数据库中数据表book里。



RecNo	book_id	book_name	item_id	identify	auto_release	is_download	order_index	description	publisher	label	privately_owned	private_token	book_password	status	editor	doc_count	comment_status	cor
Click here to define a filter																		
1	1	MinDoc演示项目	1	mindoc	0	0	0	这是一个MinDoc演示项目，该项目是由系统初始化时自动创建。			0			0	markdown	1	closed	
2	4	4 wangeditor html编辑器	1	momo	0	0	0				0			0	html	10	closed	
3	5	2 cherry markdown编辑器	2	sd	0	1	0				0			0	cherry_markdown	1	closed	
4	6	3 quill html编辑器	1	onlyoffice	0	0	0				0			0	new_html	72	closed	
5	7	1 markdown	1	oo	0	0	0				0			0	markdown	70	closed	

查阅book

当我们查阅book的时候
对应的服务端

```
// 阅读文档
func (c *DocumentController) Read() {

c.TplName = fmt.Sprintf("document/%s_read.tpl", bookResult.Theme)

data.DocIdentify = doc.Identify
```

服务端取到的内容通过 `kancloud.js` 加载到 `defult_read.tpl`

```
/***
 * 加载文档到阅读区
 * @param $url
 * @param $id
 * @param $callback
 */
function loadDocument($url, $id, $callback) {
    $.ajax({
        url: $url,
        type: "GET",
        beforeSend: function () {
```

编辑

来到view里的 `defult_read.tpl`
右上角的 `编辑` 按钮对应的地址 `http://127.0.0.1:8181/api/oo/edit/` 如下

```
                {{if gt .Member.MemberId 0}}
                {{if eq .Model.RoleId 0 1 2}}
                <div class="dropdown pull-left" style="margin-right: 10p
x;">
```

```
                    <a href="{{urlfor "DocumentController.Edit" ":key" .Model.Identify ":id" ""}}" class="btn btn-danger"><i class="fa fa-edit" aria-hidden="true"></i> {{i18n .Lang "blog.edit"}}</a>
                    {{if eq .Model.RoleId 0 1}}
                    <a href="{{urlfor "BookController.Users" ":key" .Model.Identify}}" class="btn btn-success"><i class="fa fa-user" aria-hidden="true"></i> {{i18n .Lang "blog.member"}}</a>
                    <a href="{{urlfor "BookController.Setting" ":key" .Model.Identify}}" class="btn btn-primary"><i class="fa fa-gear" aria-hidden="true"></i> {{i18n .Lang "common.setting"}}</a>
                    {{end}}
                </div>
                {{end}}
                {{end}}
```

根据 `urlfor "DocumentController.Edit" ":key" .Model.Identify ":id" ""` 其中的key是 `oo` ，即项目名称，id是空。
来到 `DocumentController.Edit`

```
// 编辑文档
func (c *DocumentController) Edit() {

    identify := c.Ctx.Input.Param(":key")

    c.TplName = fmt.Sprintf("document/%s_edit_template.tpl", bookResult.Editor)
```

identify就是项目名称。根据项目名称打开editor编辑器。
来到编辑器 `markdown_edit_template.tpl`

```
 window.editURL = "{{urlfor "DocumentController.Content" ":key" .Model.Identify ":id" ""}}";
```

点击左侧目录，就会通过 `markdown.js` 将内容取出来，服务端对应的 `DocumentController.Content`

```
    /***
     * 加载指定的文档到编辑器中
     * @param $node
     */
    window.loadDocument = function ($node) {
        var index = layer.load(1, {
            shade: [0.1, '#fff'] // 0.1 透明度的白色背景
```

```
        });

        $.get(window.editURL + $node.node.id).done(function (res) {
            layer.close(index);
```

服务端 `DocumentController.Content`

```
// 获取文档内容
func (c *DocumentController) Content() {
    c.Prepare()

    identify := c.Ctx.Input.Param(":key")
    docId, err := c.GetInt("doc_id")

    if err != nil {
        docId, _ = strconv.Atoi(c.Ctx.Input.Param(":id"))
    }
```

默认页

问题，当我们正在查看任何一页，点击编辑，立刻进入该页的编辑器，如何传递默认页的id到编辑器里的?
还是在 `kancloud.js` 中，保存到了缓存中了。然后在 `markdown.js` 加载指定id的内容到编辑器。

```
var events = function () {
    var articleOpen = function (event, $param) {
        //当打开文档时，将文档ID加入到本地缓存。
        window.sessionStorage && window.sessionStorage.setItem("MinDoc::L
astLoadDocument:" + window.book.identify, $param.$id);
        var prevState = window.history.state || {};
        if ('pushState' in history) {
            if ($param.$id) {
                prevState.$id === $param.$id || window.history.pushState
($param, $param.$id, $param.$url);
            } else {
                window.history.pushState($param, $param.$id, $param.$ur
l);
                window.history.replaceState($param, $param.$id, $param.$u
rl);
            }
        } else {
            window.location.hash = $param.$url;
```

```
        }

        initHighlighting();
        $(window).resize();

        $(".manual-right").scrollTop(0);
        //使用layer相册功能查看图片
        layer.photos({ photos: "#page-content" });
    };
```

第二天

添加setting.tpl里的froala，见上面第一段代码

添加const.go

```
package controllers

const (
    Markdown             = "markdown"
    EditorMarkdown       = "markdown"
    EditorCherryMarkdown = "cherry_markdown"
    EditorHtml           = "html"
    EditorNewHtml        = "new_html"
    EditorFroala         = "froala"
)
```

将 `html_edit_template.tpl` 拷贝一份改名为 `froala_edit_template.tpl`
将里面的引用css和js替换为froala的
再拷贝js里的 `html-editor.js` ，改名为 `froala-editor.js`

```
/***
     * 加载指定的文档到编辑器中
     * @param $node
     */
    function loadDocument($node) {
        var index = layer.load(1, {
            shade: [0.1,'#fff'] //0.1透明度的白色背景
        });

        $.get(window.editURL + $node.node.id ).done(function (res) {
            layer.close(index);
```

```
            if(res.errcode === 0){
                window.isLoad = true;

                // window.editor.txt.clear();
                // window.editor.txt.html(res.data.content);
                window.editor.html.set('');
                window.editor.events.focus();
                window.editor.html.set(res.data.content);
```

`DocumentController.go` 里的editor将Model参数传给了 `froala_edit_template.tpl` 的 `window.editURL` ，即对应的 `.Model.Identify`

```
type BookResult struct {
    Identify       string    `json:"identify"`
}

// 编辑文档
func (c *DocumentController) Edit() {
    c.Data["Model"] = bookResult
```

`froala_edit_template.tpl` 里的 `Model.Identify` 来自上面的Edit方法

```
window.editURL = "{{urlfor "DocumentController.Content" ":key" .Model.Identify ":id" ""}}";
```

bookctroller.go

```
    // if editor != EditorMarkdown && editor != EditorCherryMarkdown && editor != EditorHtml && editor != EditorNewHtml {
    if editor != EditorMarkdown && editor != EditorCherryMarkdown && editor != EditorHtml && editor != EditorNewHtml && editor != EditorFroala {
        editor = EditorMarkdown
    }
```

documentcontroller.go

```
    if name == "file" {
        // froala单图片上传
        c.Ctx.Output.JSON(result, true, false)
    } else {
```

```
        c.Ctx.Output.JSON(result2, true, false)
    }
    c.StopRun()
```

https://froala.com/wysiwyg-editor/examples/
https://froala.com/wysiwyg-editor/examples/getHTML/
https://froala.com/wysiwyg-editor/examples/custom-buttons/

9 为开源做贡献

这里记录如何为开源做贡献，经过摸索，也走过不少坑，但仍然有更多的坑等着。
所以要小心和简单，不要搞复杂了，对应我们这样的纯粹就是为了写点代码的人而言。
将项目fork到自己的git账户下后，在本机clone下来2个地方，一个地方用于开发，一个地方将更新同步过来后专门用于push到自己的git账户上，然后pull request到官方。这样保证pr的东西很清爽。

cmd真实记录

```
Administrator@DESK-XDIGIWSFLS MINGW64 /d/gowork/src/github.com/3xxx
$ git clone https://github.com/3xxx/mindoc.git
Cloning into 'mindoc'...
remote: Enumerating objects: 11960, done.
remote: Counting objects: 100% (823/823), done.
remote: Compressing objects: 100% (595/595), done.
remote: Total 11960 (delta 284), reused 675 (delta 213), pack-reused 1113
7
Receiving objects: 100% (11960/11960), 90.08 MiB | 10.69 MiB/s, done.
Resolving deltas: 100% (6172/6172), done.

Administrator@DESK-XDIGIWSFLS MINGW64 /d/gowork/src/github.com/3xxx
$ cd mindoc

Administrator@DESK-XDIGIWSFLS MINGW64 /d/gowork/src/github.com/3xxx/mindo
c (master)
$ git remote add upstream https://github.com/mindoc-org/mindoc.git

Administrator@DESK-XDIGIWSFLS MINGW64 /d/gowork/src/github.com/3xxx/mindo
c (master)
$ git remote -v
origin  https://github.com/3xxx/mindoc.git (fetch)
origin  https://github.com/3xxx/mindoc.git (push)
upstream        https://github.com/mindoc-org/mindoc.git (fetch)
upstream        https://github.com/mindoc-org/mindoc.git (push)

Administrator@DESK-XDIGIWSFLS MINGW64 /d/gowork/src/github.com/3xxx/mindo
c (master)
$ git fetch upstream master
From https://github.com/mindoc-org/mindoc
 * branch            master     -> FETCH_HEAD
 * [new branch]      master     -> upstream/master
```

```
以后每次上传前，从这里开始：这里先更新一下
意思是从官方拉取更新（如果有更新），而不是从你的git上拉取更新
Administrator@DESK-XDIGIWSFLS MINGW64 /d/gowork/src/github.com/3xxx/mindo
c (master)
$ git pull upstream master
remote: Enumerating objects: 1, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 1 (delta 0), reused 1 (delta 0), pack-reused 0
Unpacking objects: 100% (1/1), 859 bytes | 859.00 KiB/s, done.
From https://github.com/mindoc-org/mindoc
 * branch            master     -> FETCH_HEAD
   397dced..db620be  master     -> upstream/master
hint: Waiting for your editor to close the file...
/usr/bin/bash: line 1: q: command not found

shell returned 127

Press ENTER or type command to continue
Merge made by the 'ort' strategy.

开始将本机更新代码添加到更新队列
Administrator@DESK-XDIGIWSFLS MINGW64 /d/gowork/src/github.com/3xxx/mindo
c (master)
$ git add .

Administrator@DESK-XDIGIWSFLS MINGW64 /d/gowork/src/github.com/3xxx/mindo
c (master)
$ git commit -m "update images upload"
[master af88388] update images upload
 2 files changed, 306 insertions(+), 260 deletions(-)

Administrator@DESK-XDIGIWSFLS MINGW64 /d/gowork/src/github.com/3xxx/mindo
c (master)
$ git push
warning: ----------------- SECURITY WARNING ----------------
warning: | TLS certificate verification has been disabled! |
warning: ---------------------------------------------------
warning: HTTPS connections may not be secure. See https://aka.ms/gcm/tlsv
erify for more information.
warning: ----------------- SECURITY WARNING ----------------
warning: | TLS certificate verification has been disabled! |
warning: ---------------------------------------------------
warning: HTTPS connections may not be secure. See https://aka.ms/gcm/tlsv
erify for more information.
```

```
Enumerating objects: 17, done.
Counting objects: 100% (17/17), done.
Delta compression using up to 20 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (9/9), 3.21 KiB | 3.21 MiB/s, done.
Total 9 (delta 7), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (7/7), completed with 7 local objects.
To https://github.com/3xxx/mindoc.git
   397dced..af88388  master -> master
```

1 win下编译mindoc

修改conf里
注释掉mysql
取消sqlite注释
拷贝一个sqlite数据库到database文件夹里，更名为mindoc.db
cmd进入mindoc文件夹
因为用sqlite需要cgo，go env -w CGO_ENABLED=1
https://blog.csdn.net/qq_43625649/article/details/134488353

D:\gowork>cd src\github.com\mindoc
D:\gowork\src\github.com\mindoc>bee run

2 Fork 这个项目

点一下fork按钮即可。

3 Clone 已经 fork 的项目

当你 fork 一个项目之后，你需要把它克隆到你的电脑上，这样你才能开始工作。
要 clone 这个项目，你先打开你自己的 GitHub 主页，找到 fork 过来的项目，打开后点击右上角的 “Clone or download” 按钮，得到复制的地址。
GitHub 为克隆项目提供了两种传输协议： HTTPS 和 SSH 。关于这个主题的更多内容可以看这里。这里假设你决定使用 HTTPS 。
当你复制了项目 URL，你可以在 Git 客户端或者 shell 里 clone 项目：

$ git clone https://github.com/mindoc-org/mindoc.git
Clone 一个项目会在你的硬盘上创建一个文件夹，里面有项目的所有文件，还有跟踪文件变化的 git 文件夹。
特别注意：

1. clone到本地的项目文件夹作为将来提交的文件夹，因此要保持干净，不要在这里面进行开发调试。
   本地开发调试要另外再clone这个项目，完成开发后把个别修改的文件同步到前面clone的文件夹里，然后在它里面进行git push等提交代码操作。
2. 提交到自己github上fork的项目里后，再进行pull request操作。

4 设置克隆过来的项目

进入克隆过来的项目文件夹，将原来项目的 URL （官方）添加到你的本地代码仓库，这样你就可以随时从原来的项目 pull 最新的修改:

$ git remote add upstream url
我用 upstream (上游、官方) 作为远程仓库的名字，这是 GitHub 的风格，但是你可以用任何名字。

现在远程仓库列表是这样的:

Administrator@DESK-XDIGIWSFLS MINGW64 /d/gowork/src/github.com/mindoc (master)
$ git remote add upstream https://github.com/mindoc-org/mindoc.git

Administrator@DESK-XDIGIWSFLS MINGW64 /d/gowork/src/github.com/mindoc (master)
$ git remote -v
origin https://github.com/engineercms/mindoc.git (fetch)
origin https://github.com/engineercms/mindoc.git (push)
upstream https://github.com/mindoc-org/mindoc.git (fetch)
upstream https://github.com/mindoc-org/mindoc.git (push)

这里创建了一个叫做upstream（官方）的与源代码origin关联,关联之后，就可以同步源代码库的其他分支到自己的仓库了
在你把自己的修改 push 到你的 fork 之前，从上游（官方）拉回一次新的变动。

Administrator@DESK-XDIGIWSFLS MINGW64 /d/gowork/src/github.com/mindoc (master)
$ git fetch upstream master
From https://github.com/mindoc-org/mindoc

- branch master -> FETCH_HEAD
- [new branch] master -> upstream/master

Administrator@DESK-XDIGIWSFLS MINGW64 /d/gowork/src/github.com/mindoc (master)
$ git pull upstream master
From https://github.com/mindoc-org/mindoc

- branch master -> FETCH_HEAD

  Already up to date.

5 推送

Administrator@DESK-XDIGIWSFLS MINGW64 /d/gowork/src/github.com/mindoc (master)

$ git add .

warning: in the working copy of 'conf/app.conf.example', LF will be replaced by CRLF the next time Git touches it

Administrator@DESK-XDIGIWSFLS MINGW64 /d/gowork/src/github.com/mindoc (master)

$ git commit -m "add prev&next in default read.tpl"

[master 2988fab] add prev&next in default read.tpl
11 files changed, 1776 insertions(+), 95 deletions(-)
create mode 100644 conf/app.conf.example
delete mode 100644 uploads/202404/cover_17c7b2c6745688e0.png
create mode 100644 uploads/202404/cover_17c8260e47e06f18_small.png
create mode 100644 uploads/202404/cover_17c8286be57a8600_small.png
create mode 100644 uploads/202404/cover_17c829abdef5fdc8.png
create mode 100644 uploads/books/2/book.zip
create mode 100644 views/document/test.tpl
delete mode 100644 "win\344\270\213\347\274\226\350\257\221.md"

Administrator@DESK-XDIGIWSFLS MINGW64 /d/gowork/src/github.com/mindoc (master)

$ git push

warning: ———————— SECURITY WARNING ————————-
warning: | TLS certificate verification has been disabled! |
warning: ————————————————————
warning: HTTPS connections may not be secure. See https://aka.ms/gcm/tlsverify for more information.
warning: ———————— SECURITY WARNING ————————-
warning: | TLS certificate verification has been disabled! |
warning: ————————————————————
warning: HTTPS connections may not be secure. See https://aka.ms/gcm/tlsverify for more information.
Enumerating objects: 33, done.
Counting objects: 100% (33/33), done.
Delta compression using up to 20 threads

Compressing objects: 100% (19/19), done.
Writing objects: 100% (21/21), 249.03 KiB | 27.67 MiB/s, done.
Total 21 (delta 11), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (11/11), completed with 11 local objects.
To https://github.com/engineercms/mindoc.git
6054b5f..2988fab master -> master

1. 问题一

甲乙两个程序员正在合作开发一款程序，甲晚上10点push了自己编辑的test.txt文档到GitHub。乙11点也想要push自己编辑过后的test.txt文档到GitHub,因为乙本地仓库不是最新，乙push时会报错，乙如果使用git pull拉取最新版本，那自己修改的东西丢失了怎么办?
解决办法:
先把自己修改好的代码存放在缓存里，等最新代码拉取下来以后再恢复缓存里的自己修改的代码,乙再push。
前提是两人不要修改相同的地方，可能会产生冲突!
步骤如下:

```
git stash save "这里是注释"
git pull
git stash pop
git stash list
git stash show
```

2. 问题二

乙在上个问题背景时不知道甲做了提交，直接commit了自己的test.txt文件，想取消自己的commit怎么办?
解决办法:

```
git reset --soft HEAD^
```

原文链接：https://blog.csdn.net/m0_61502213/article/details/120620168

git-拉取代码不覆盖本地修改的代码

1 先将本地代码放在暂存区:

```
git stash
```

2 将远程gitlab上代码拉取下来：

```
git pull
```

3 将1中暂存区的代码放回本地：

```
git stash pop
```

4 接下来将代码合并到master：

4.1 提交到远程库：

方法一：提交tracked的修改

```
git commit -am 'comment code1'
git push
```

方法二：提交未tracked的修改

```
git add -u
git commit -m 'comment code1'
git push
```

4.2 codereview:

```
arc diff origin/master
```

4.3 合并到master：

```
arc land --onto master
```

其他：

```
git add -u <==> git add –update
```

提交所有被删除和修改的文件到数据暂存区

```
git add .
```

提交所有修改的和新建的数据暂存区

```
git add -A <==>git add –all
```

提交所有被删除、被替换、被修改和新增的文件到数据暂存区

【gitHub】在提PR（pull request）时提交指定的commit

一、背景
今天在提PR的时候，发现以前一些旧的commit也一起被提交上去了，这样比较不友好，在网上研究了一些方法，在这里总结记录一下。

二、方法
对于方法博主研究后有两种，一种是通过版本非强制回退，然后再重新commit一次，这样就能够合并之前的所有commit。另外一种是通过使用cherry-pick选择commit进行提交。博主推荐使用第二种。

2.2.使用cherry-pick进行提交
cherry-pick是在新分支下选择本地分支的commit记录进行合并的操作，因此只需要在本地新建一个上游仓库（注意不是远程仓库）的本地分支，再合并想要提交的commit即可。

首先，配置上游仓库地址并同步。

```
git remove add upstream XXX.git // XXX.git是上游仓库地址
git fetch
```

在github上或者在本地运行git log选取commit记录。
注意：需要选择上次PR之后的除了merge操作的所有commit的id。

在本地从上游仓库中拉取新的分支。
注意：upstream是上游仓库，在第一步配置过；origin是自己的远程仓库；main是主分支名。

```
git checkout -b new_branch upstream/main
```

在new_branch的分支上cherry-pick所有commit记录。
注意：要选择所有未提交PR的所有commit，这样才不会出错。

```
git cherry-pick commit_id1 commit_id2 commit_id3 ......
```

将新分支推送到远程仓库。

```
git push origin new_branch
```

在github提PR的时候选择新推送的分支即可。

PR被接受后，可选择删除新建立的本地分支和远程分支。

```
git branch -D new_branch
```
// 在main分支删除new_branch分支

```
git push origin --delete new_branch
```
// 删除远程仓库分支

版权声明：本文为博主原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接和本声明。
原文链接：https://blog.csdn.net/wuchus/article/details/126018674

git 报错

在使用 Git 进行代码管理 的过程中，经常会遇到各种各样的问题，其中之一就是在执行 git clone 或 git pull 等操作时出现 “fatal: unable to access ‘https://github.com/…/.git’: Recv failure Connection was reset” 的报错。这个问题通常是由网络连接 问题或代理设置不正确导致的。在我的个人使用经验中，我亲自尝试了两种方法，它们都能够有效地解决这个报错。

方法一：取消代理设置
这是最常见的解决方法之一，通过在终端执行以下命令，可以取消 Git 的代理设置：

git config –global –unset http.proxy
git config –global –unset https.proxy

版权声明：本文为CSDN博主「IT·陈寒」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。
原文链接：https://blog.csdn.net/qq_43546721/article/details/139506583

10 添加目录访问密码

static\js\markdown.js

```
    /**
     * 文档目录树
     */
    $("#sidebar").jstree({
        'plugins': ["wholerow", "types", 'dnd', 'contextmenu'],
        "types": {
            "default": {
                "icon": false  // 删除默认图标
            }
        },
        'core': {
            'check_callback': true,
            "multiple": false,
            'animation': 0,
            "data": window.documentCategory
        },
        "contextmenu": {
            show_at_node: false,
            select_node: false,
            "items": {
                "添加文档": {
                    "separator_before": false,
                    "separator_after": true,
                    "_disabled": false,
                    "label": window.editormdLocales[window.lang].addDo
c,//"添加文档",
                    "icon": "fa fa-plus",
                    "action": function (data) {
                        var inst = $.jstree.reference(data.reference),
                            node = inst.get_node(data.reference);
                        openCreateCatalogDialog(node);
                    }
                },
                "编辑": {
                    "separator_before": false,
                    "separator_after": true,
                    "_disabled": false,
                    "label": window.editormdLocales[window.lang].edit,
                    "icon": "fa fa-edit",
                    "action": function (data) {
```

```
                        var inst = $.jstree.reference(data.reference);
                        var node = inst.get_node(data.reference);
                        openEditCatalogDialog(node);
                    }
                },
                "删除": {
                    "separator_before": false,
                    "separator_after": true,
                    "_disabled": false,
                    "label": window.editormdLocales[window.lang].delete,
                    "icon": "fa fa-trash-o",
                    "action": function (data) {
                        var inst = $.jstree.reference(data.reference);
                        var node = inst.get_node(data.reference);
                        openDeleteDocumentDialog(node);
                    }
                }
            }
        }
    }).on("ready.jstree", function () {
```

直接利用添加文档的弹框和保存功能即可，编辑也是利用了添加文档的模态框和保存功能。
所以接下来就是数据库中添加2个字段，模态框添加2行输入，一行是密码，一行是密码提示
服务端的文档保存要添加这2个字段，数据库建立要添加这2个字段，
目录显示要判断这个目录是否有密码，有则加个星号
点击这个目录阅读文章要弹框输入密码，显示密码提示信息。
密码正确了，才从服务端获取文档。
如果是其他模版，比如cherry_read.tpl，是否需要更有关改代码?

11 集成onlyoffice

https://github.com/chengked/mindoc 项目地址

onlyoffice加密相关的在models.OoModel.go里，包括配置api及jwt token