# GoWeb

## Request解析

r \*http.Request

r.ParseForm() //解析

输入：

<http://localhost:9090/list?url_long=111&url_long=222&p1=1&p2=2&p3=3>

### r.Form

输出：

map[url\_long:[111 222] p1:[1] p2:[2] p3:[3]]

### r.URL.Path

/list

### r.URL.Scheme

### r.Method

GET

POST

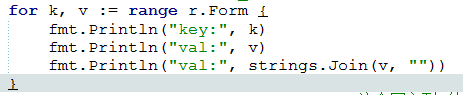
### 打印输入参数

fmt.Println(**r.Form["url\_long"]**)

fmt.Println(**r.Form["p1"]**)

r.**FormValue**("username") //这个不需要提前调用r.ParseForm()

### 遍历输入参数



输出：

key: url\_long

val: [111 222]

val: 111222

key: p1

val: [1]

val: 1

key: p2

val: [2]

val: 2

key: p3

val: [3]

val: 3

## 表单

### 下拉菜单

<select name="fruit">

<option value="apple">apple</option>

<option value="pear">pear</option>

<option value="banane">banane</option>

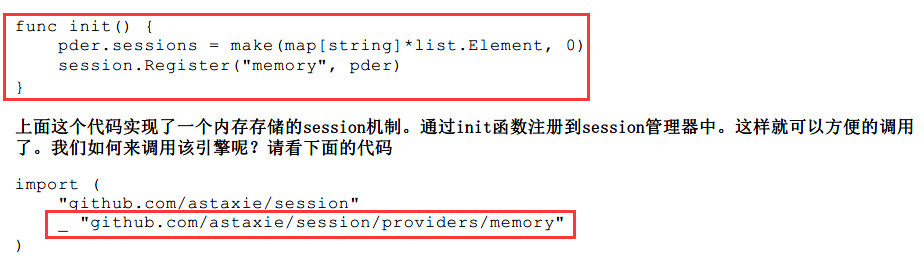
</select>

### 单选按钮

<input type="radio" name="gender" value="1">男

<input type="radio" name="gender" value="2">女

## session



### 预防session劫持



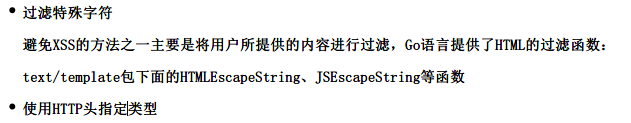
**上面两个手段的组合可以在实践中消除session劫持的风险，一方面，由于sessionID频繁改变，使攻击者难有机会获取有效的sessionID；另一方面，因为sessionID只能在cookie中传递，然后设置了httponly，所以基于URL攻击的可能性为零，同时被XSS获取sessionID也不可能。最后，由于我们还设置了MaxAge=0，这样就相当于session cookie不会留在浏览器的历史记录里面。**

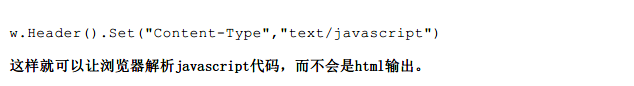
## 安全与加密

### 预防CSRF攻击 (跨站请求伪造)

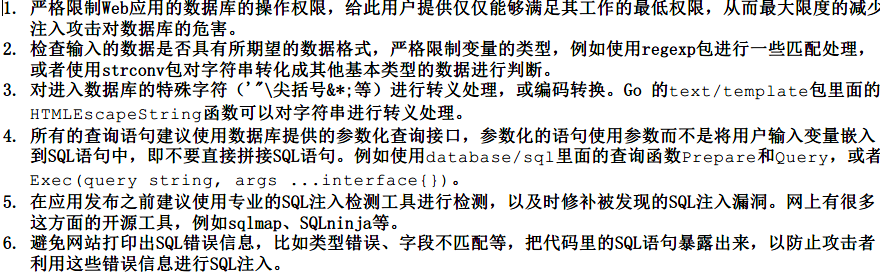


### 避免XSS攻击 (跨站脚本攻击)





### 避免SQL注入



## json

b := []byte(`{"Name":"Wednesday","Age":6,"Parents":["Gomez","Morticia"]}`)

如果在我们不知道他的结构的情况下，我们把他解析到interface{}里面

var f interface{}

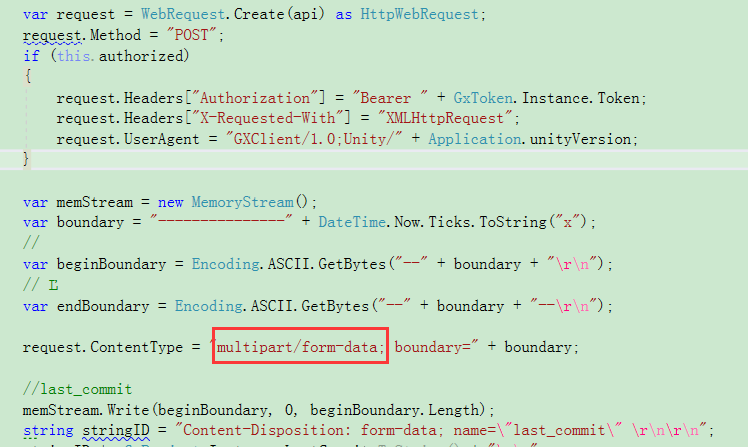
err := json.Unmarshal(b, &f)

## beego

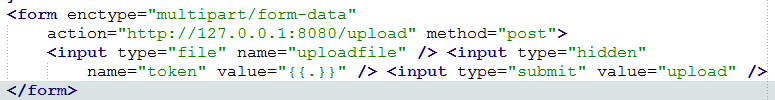
<https://github.com/astaxie/beego>

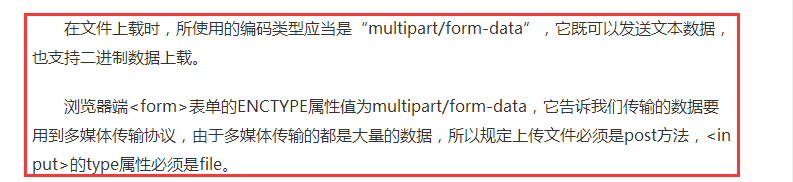
## 表单提交 （POST）

### GXPack 也用的multipart/form-data



### Multipart/form-data

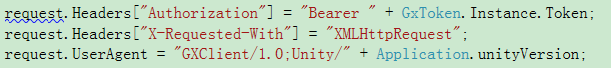




### application/x-www-form-urlencoded (form.enctype的缺省值？)

## Get

### GXPack



### Iris

## Beego

<https://beego.me/quickstart>

https://github.com/astaxie/beego

## Iris

API:

app.RegisterView(iris.HTML("./views", ".html").Reload(true))

app.OnErrorCode(iris.StatusInternalServerError, func(ctx iris.Context) {

// .Values are used to communicate between handlers, middleware.

errMessage := ctx.Values().GetString("error")

if errMessage != "" {

ctx.Writef("Internal server error: %s", errMessage)

return

}

ctx.Writef("(Unexpected) internal server error")

})

### app.Post("/upload", func(ctx iris.Context) {

app.Post("/decode", func(ctx iris.Context) {

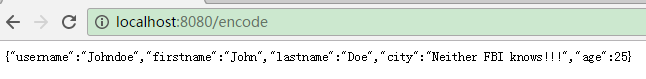
var user User

ctx.ReadJSON(&user)

ctx.Writef("%s %s is %d years old and comes from %s", user.Firstname, user.Lastname, user.Age, user.City)

})

### Json输出



app.Get("/encode", **func**(ctx iris.Context) {

doe := User{

Username: "Johndoe",

Firstname: "John",

Lastname: "Doe",

City: "Neither FBI knows!!!",

Age: 25,

}

ctx.JSON(doe)

})

# Go

## Go语法

make(map[string]Provide)

make([]byte, 32)

### defer

defer定义的语句会在最后执行。

使用场合：

及时释放资源

e.g.

pder.lock.Lock()

defer pder.lock.Unlock()

### map

//interface{} key

//interface{} value

v := make(map[interface{}]interface{}, 0)

### struct

type Server struct {

ServerName string

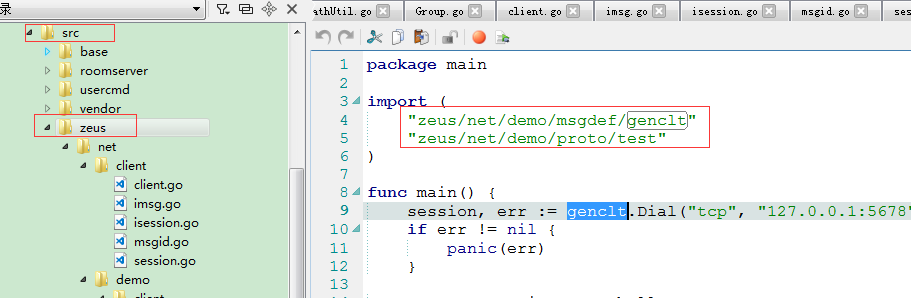
ServerIP string

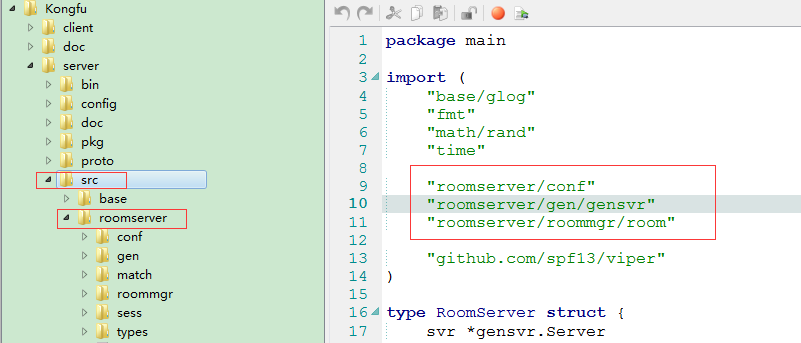
}

## 路径



### 文件路径引用

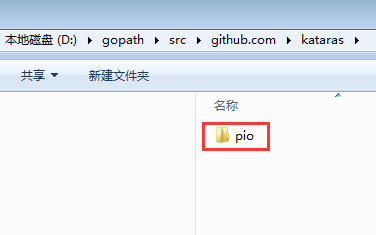




## Go Get

go get github.com/kataras/pio

路径：



# 最好的6个Go语言Web框架

<https://blog.csdn.net/dev_csdn/article/details/78740990>

Beego: 一个Go语言下开源的，高性能Web框架   
\* <https://github.com/astaxie/beego>   
\* [https://beego.me](https://beego.me/)

Buffalo: 一个Go语言下快速Web开发框架   
\* <https://github.com/gobuffalo/buffalo>   
\* [https://gobuffalo.io](https://gobuffalo.io/)

Echo: 一个高性能，极简的Web框架   
\* <https://github.com/labstack/echo>   
\* [https://echo.labstack.com](https://echo.labstack.com/)

Gin: 一个Go语言写的HTTP Web框架。它提供了Martini风格的API并有更好的性能。   
\* <https://github.com/gin-gonic/gin>   
\* <https://gin-gonic.github.io/gin>

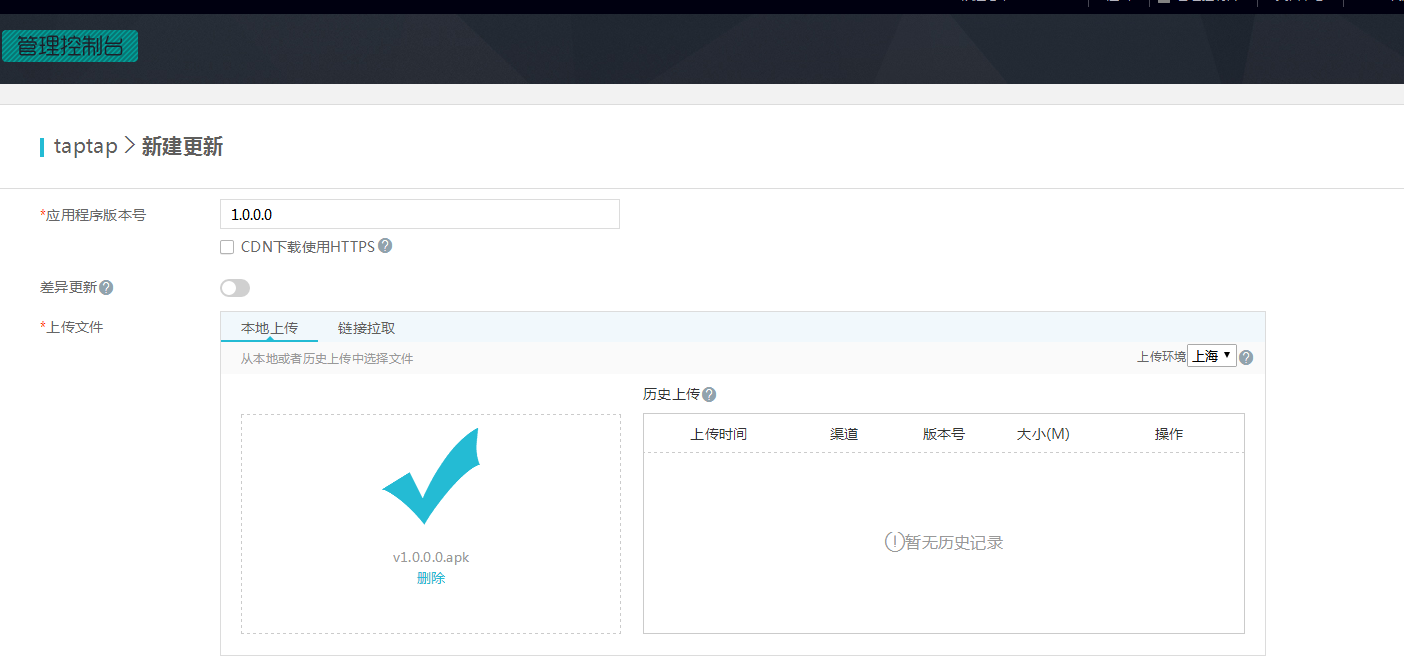
Iris: 目前发展最快的Go Web框架。提供完整的MVC功能并且面向未来。   
\* <https://github.com/kataras/iris>   
\* [https://iris-go.com](https://iris-go.com/)

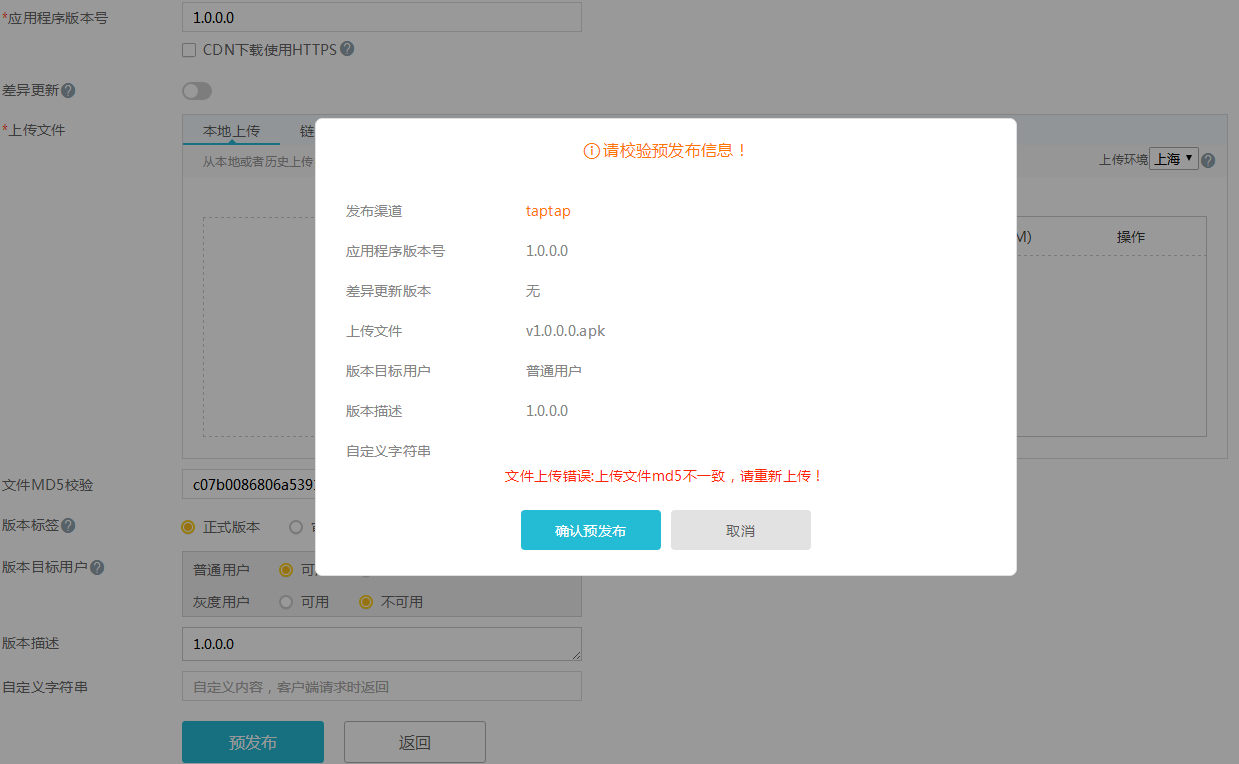
Revel: 一个高生产率，全栈Go语言的Web框架。   
\* <https://github.com/revel/revel>   
\* [https://revel.github.io](https://revel.github.io/)

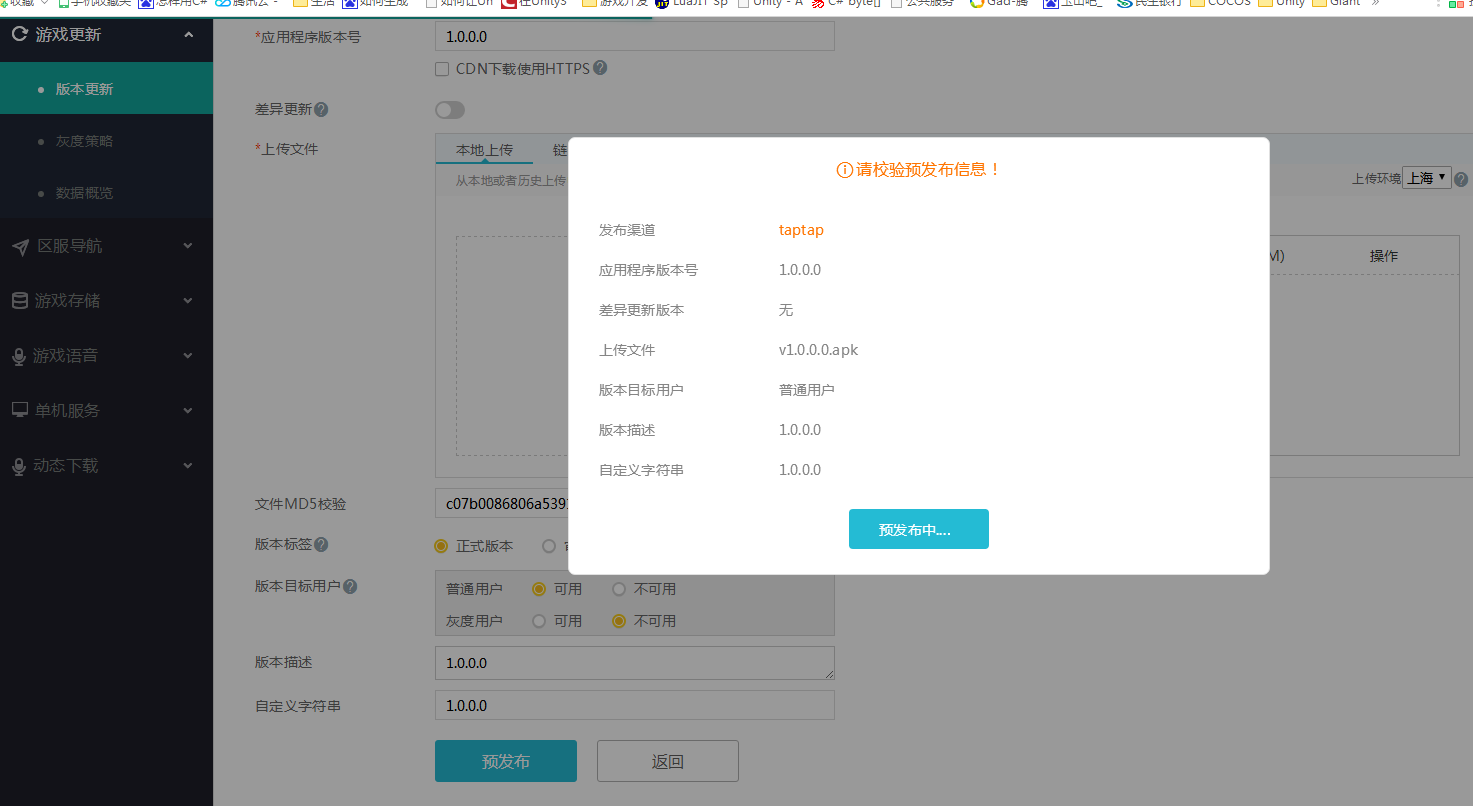
# dolphin

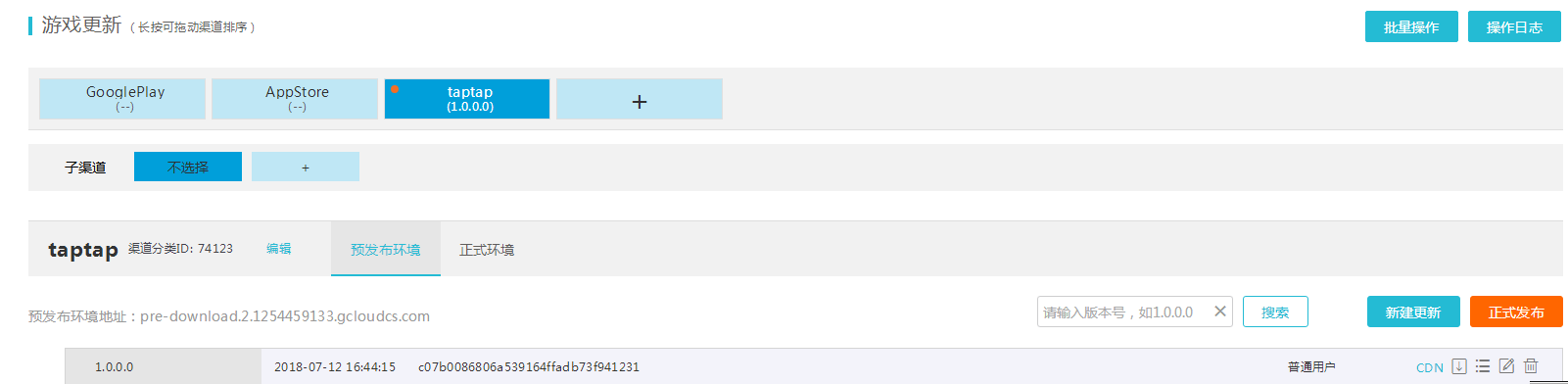
## 上传

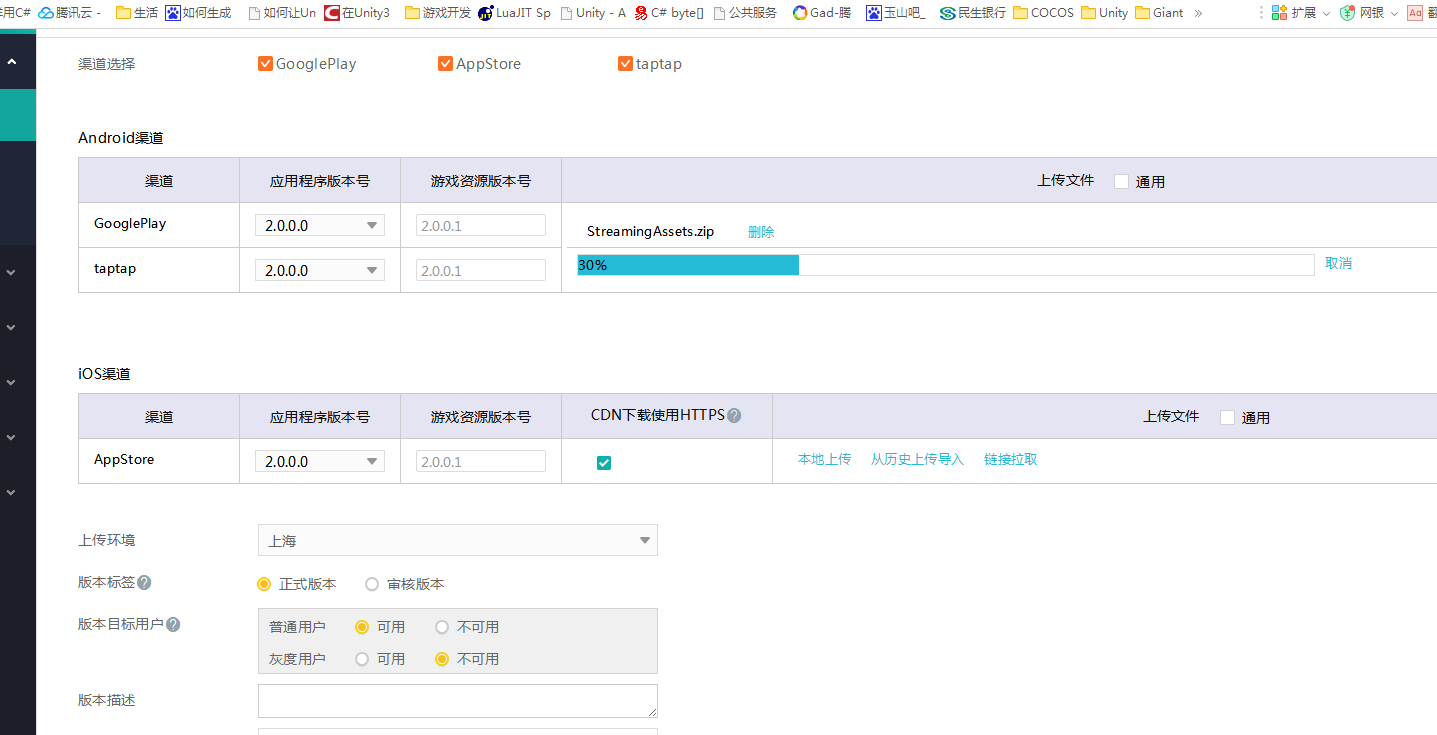












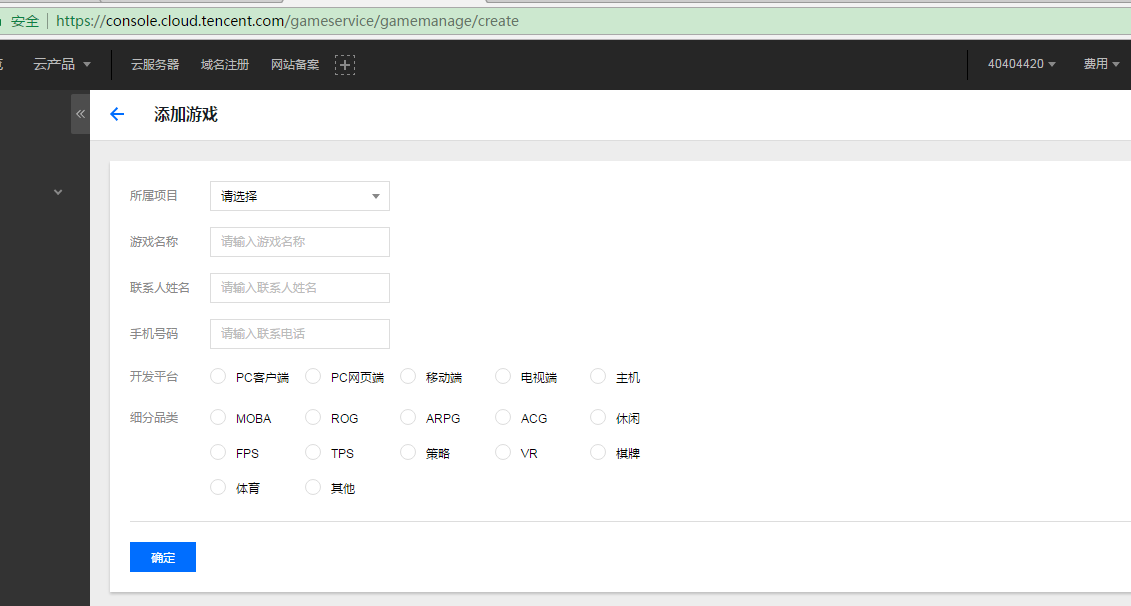
## 资源更新

每次打包的资源必须是游戏对应的全量资源，在更新的过程中，dolphin的sdk会自动和玩家本地的资源对比，下载差异资源； 以打包到zip文件中的单个文件为单位做差异，所以单个资源文件不应过大，避免大文件少量数据区修改导致大文件下载； 资源更新只做文件替换和增加，不做文件删除，游戏应避免和删除相关的游戏逻辑；

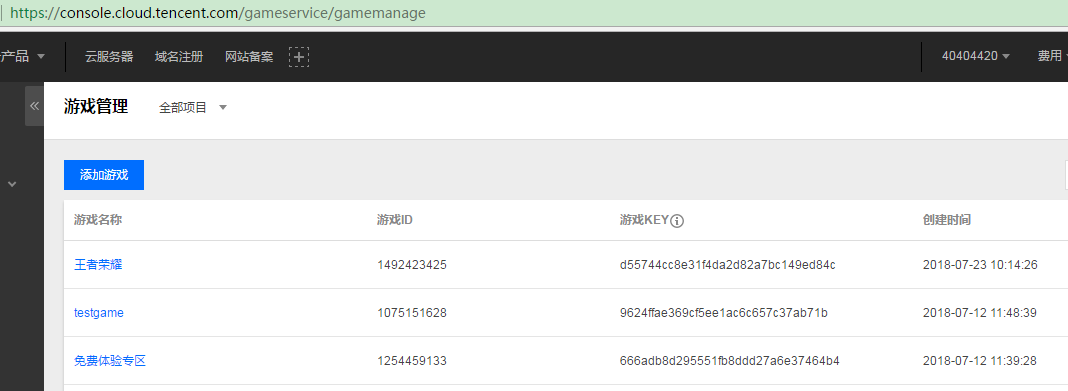
## 各种界面：

## 游戏管理

### 添加游戏

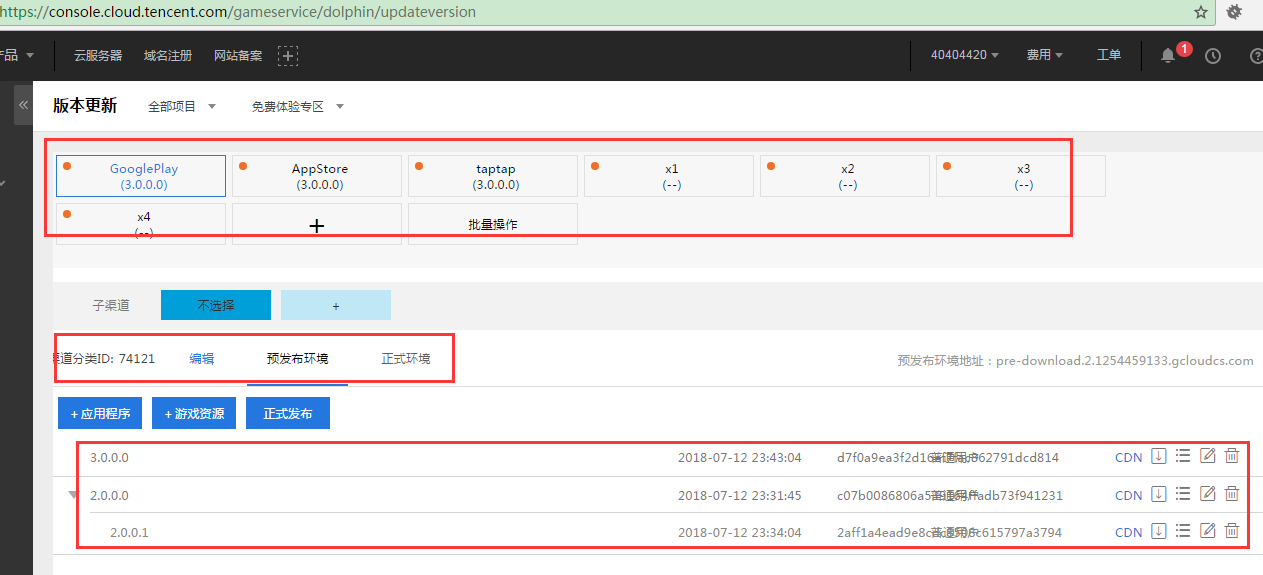


### 游戏列表

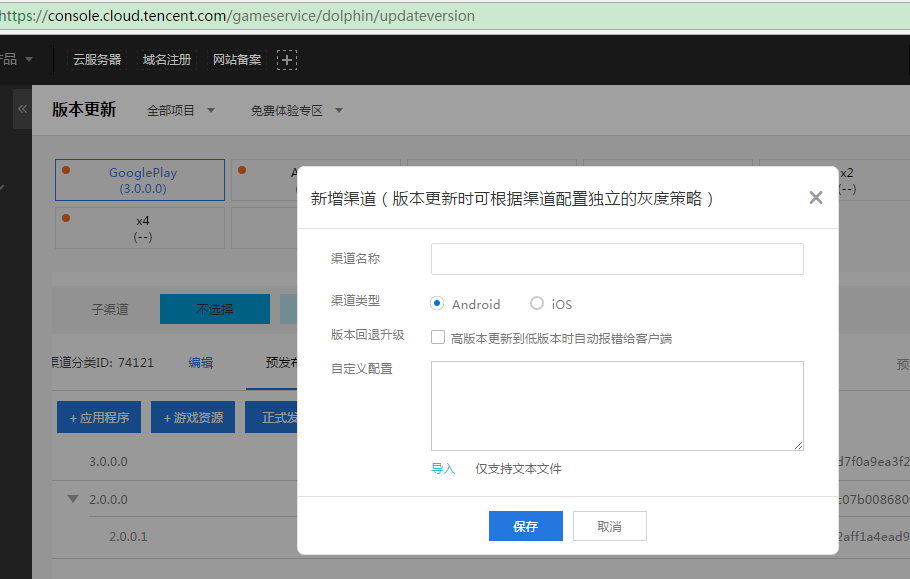


## 版本管理

### 历史版本

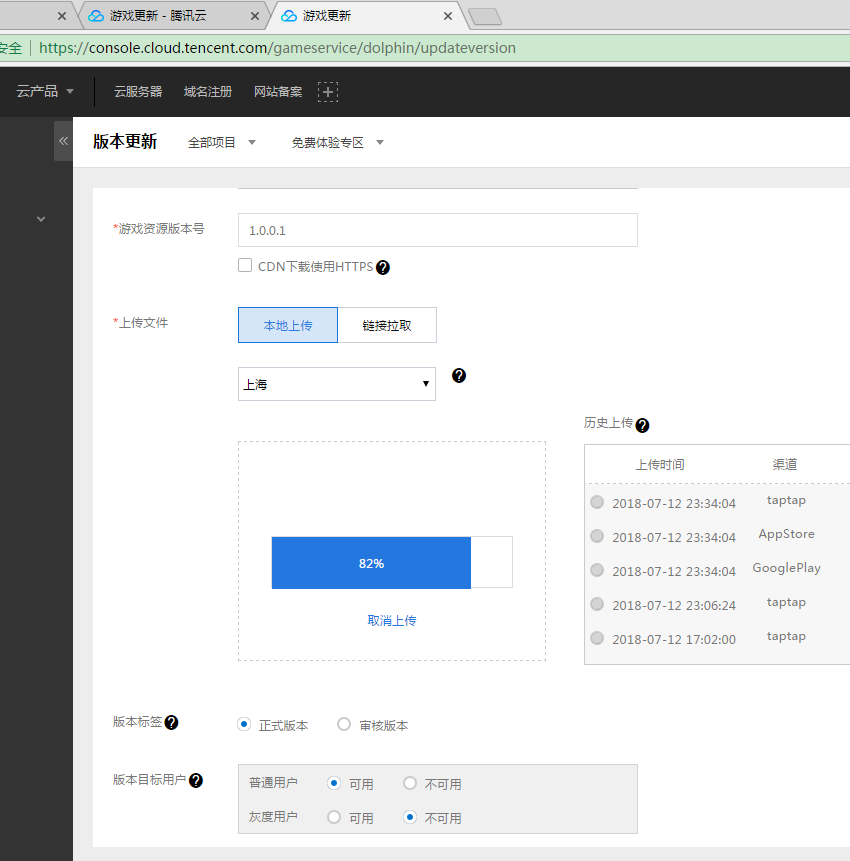


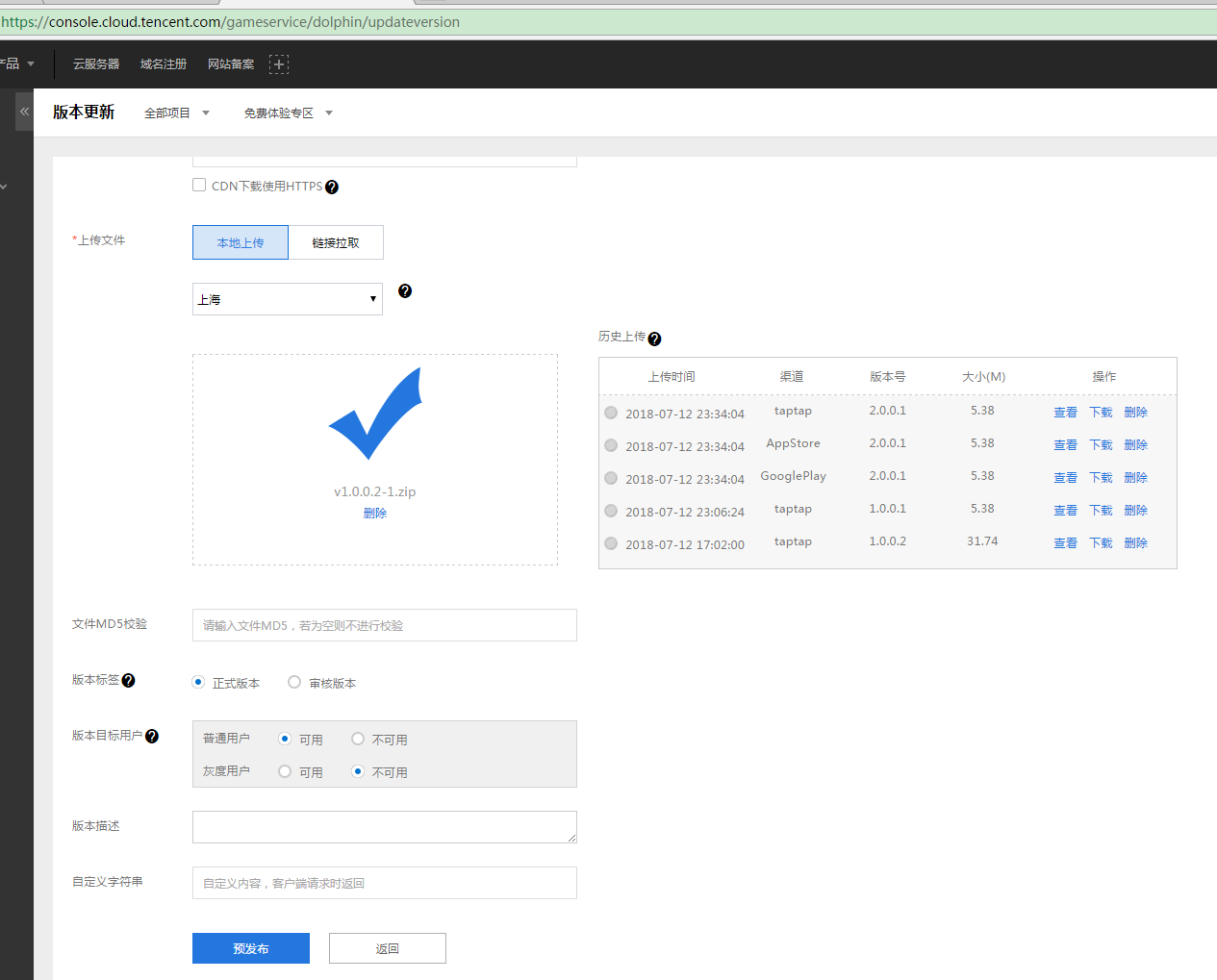
### 新增渠道





### 发布





### cdn选择



### 下载地址



### 资源更新一定是基于指定大版本的



### 资源更新必须是zip格式文件



### 预发布资源



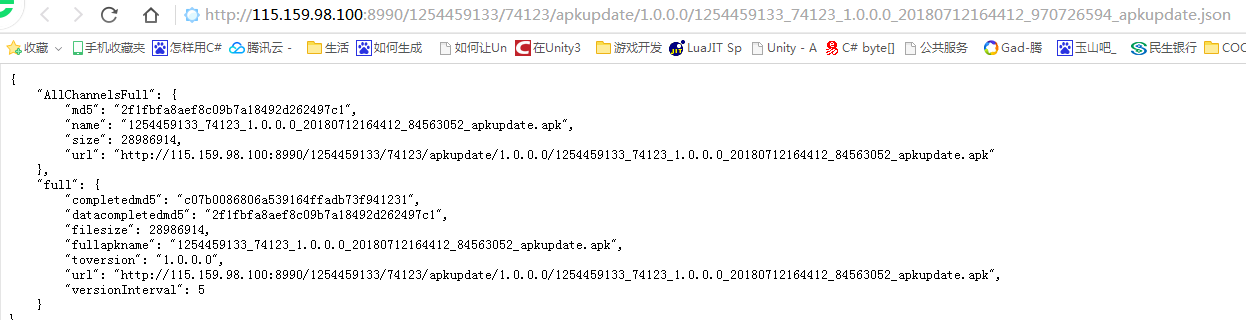
### 操作日志



## 正式发布



## 游戏版本信息



{

## 地址

预发布

<http://115.159.98.100:8990/1254459133/74123/apkupdate/1.0.0.0/1254459133_74123_1.0.0.0_20180712164412_84563052_apkupdate.apk>

<http://115.159.98.100:8990/1254459133/74123/apkupdate/1.0.0.0/1254459133_74123_1.0.0.0_20180712164412_84563052_apkupdate.apk>

http://115.159.98.100:8990/1254459133/74123/apkupdate/3.0.0.0/1254459133\_**74123**\_3.0.0.0\_20180712234258\_1234620793\_apkupdate.apk

[http://115.159.98.100:8990/1254459133/74121/**apkupdate/**3.0.0.0/1254459133\_**74121**\_3.0.0.0\_20180712234259\_36897268\_apkupdate.apk](http://115.159.98.100:8990/1254459133/74121/apkupdate/3.0.0.0/1254459133_74121_3.0.0.0_20180712234259_36897268_apkupdate.apk)

http://115.159.98.100:8990/1254459133/74121/**cures**/2.0.0.1/1254459133\_74121\_2.0.0.1\_20180712233402\_733684003\_cures.zip

http://115.159.98.100:8990/1254459133/74123/apkupdate/1.0.0.0/1254459133\_74123\_1.0.0.0\_20180712164412\_84563052\_apkupdate.apk

<http://115.159.98.100:8990/1254459133/74123/cures/1.0.0.1/1254459133_74123_1.0.0.1_20180712230621_245860808_cures.zip>

<http://115.159.98.100:8990/1254459133/74123/cures/1.0.0.1/1254459133_74123_1.0.0.1_20180712230621_245860808_cures.zip>

http://115.159.98.100:8990/1254459133/74121/cures/2.0.0.1/1254459133\_74121\_2.0.0.1\_20180712233402\_733684003\_cures.zip

正式发布

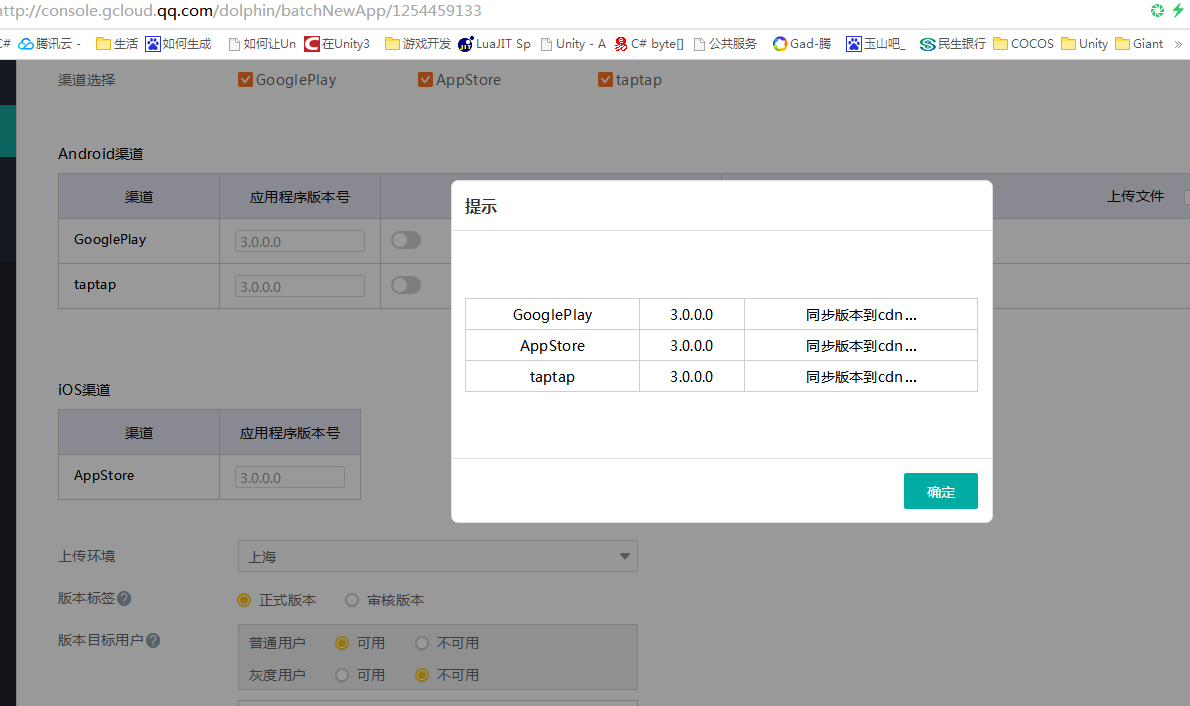
<http://115.159.98.100:8990/1254459133/74123/apkupdate/1.0.0.0/1254459133_74123_1.0.0.0_20180712164412_84563052_apkupdate.apk>

<http://115.159.98.100:8990/1254459133/74123/apkupdate/1.0.0.0/1254459133_74123_1.0.0.0_20180712164412_84563052_apkupdate.apk>

<http://115.159.98.100:8990/1254459133/74121/cures/2.0.0.1/1254459133_74121_2.0.0.1_20180712233402_733684003_cures.zip>

http://115.159.98.100:8990/1254459133/74121/cures/2.0.0.1/1254459133\_74121\_2.0.0.1\_20180712233402\_733684003\_cures.zip

## 同步到CDN：--》同步到预发布环境



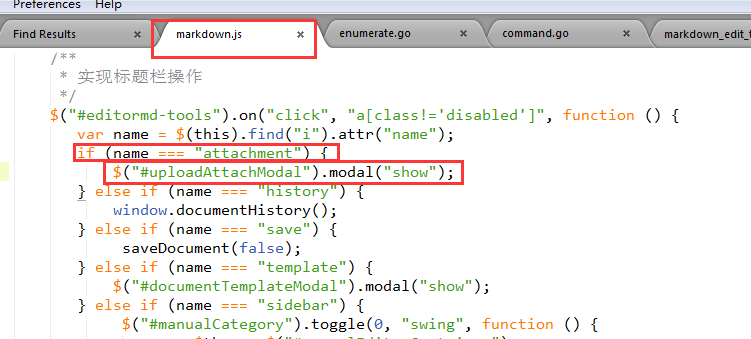
# JS



悬浮提示



点击触发的操作:



## 按钮链接？ ==============》

### href=”javascript”

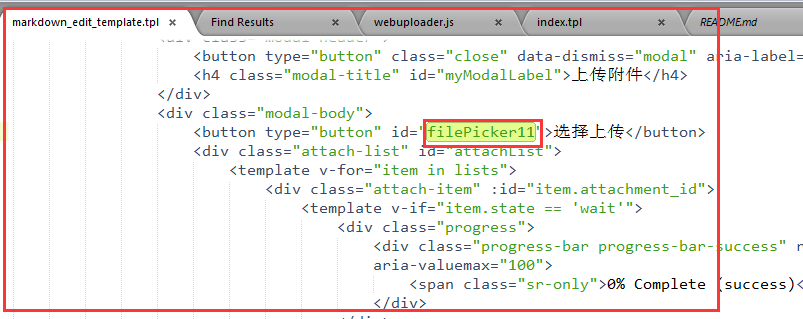
<a href="javascript:;" data-toggle="tooltip" data-title="附件"><i class="fa fa-paperclip item" aria-hidden="true" name="attachment"></i></a>

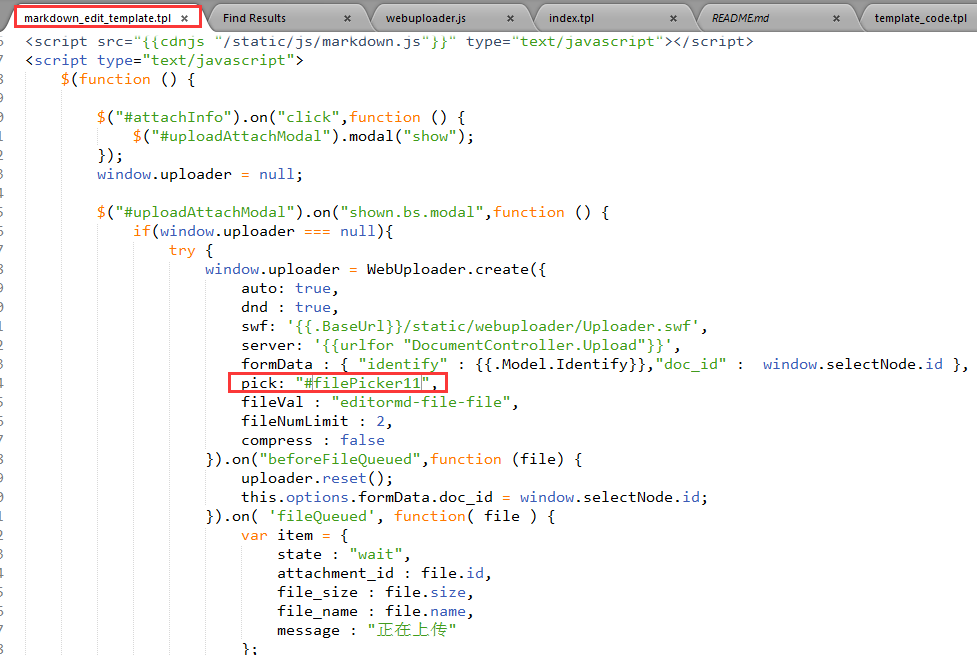
### button🡪data-toggle🡪data-target

<button type="button" data-toggle="modal" data-target="#uploadAttachModal" class="btn btn-success btn-sm pull-right">添加版本</button>

### button🡪id?

<button type="button" id="filePicker">选择上传</button>

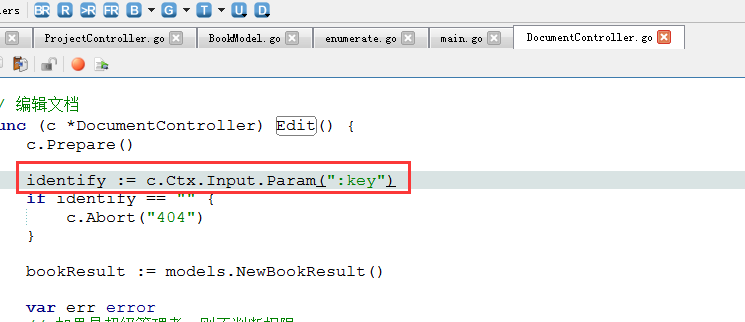




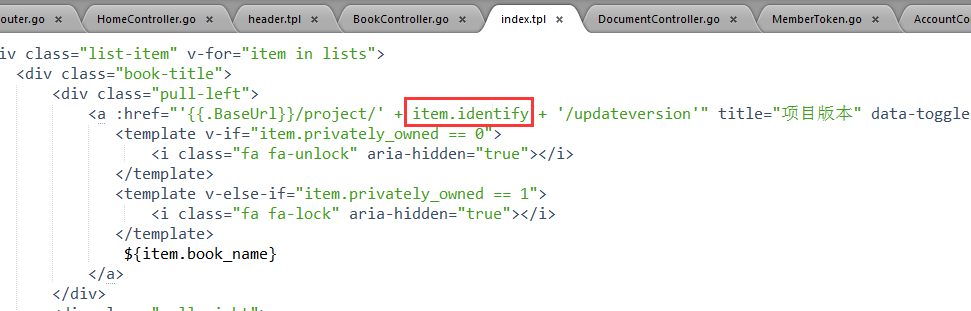
# Q&A

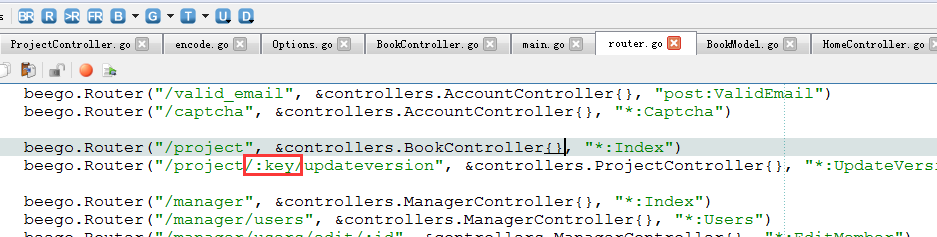
## 数据传递

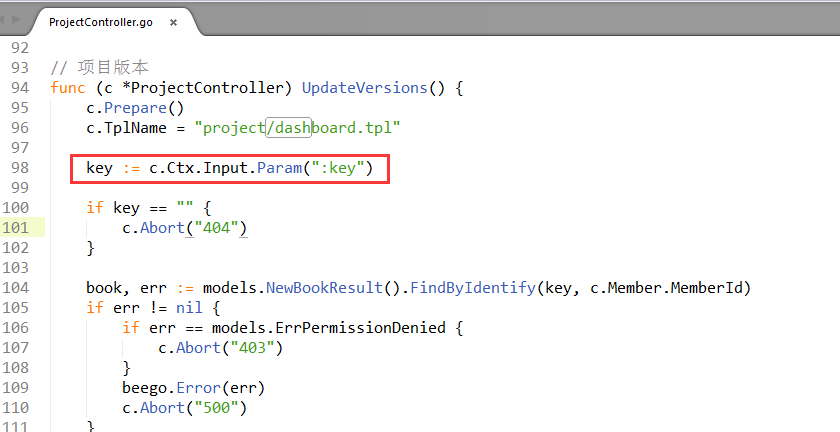
### 解析路径上的参数



### Tpl—>Controller （href给路由路径）







<a href="{{urlfor "DocumentController.Edit" ":key" .Model.Identify ":id" ""}}" class="btn btn-default btn-sm pull-right" target="\_blank"><i class="fa fa-edit" aria-hidden="true"></i> 编辑</a>

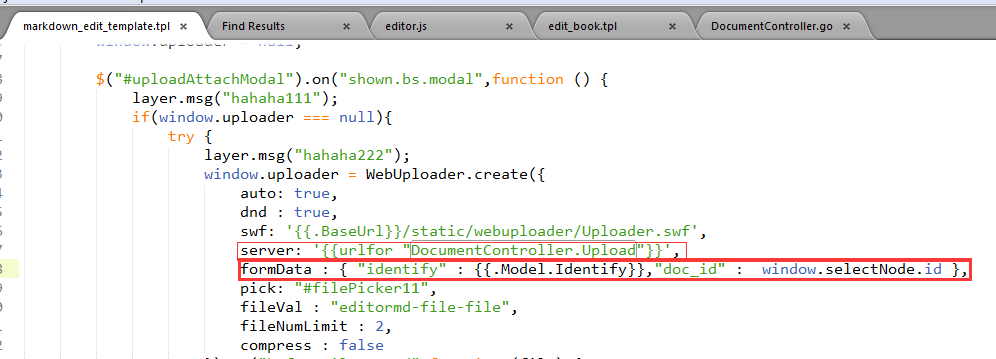
### Tpl—>Controller （post传参）

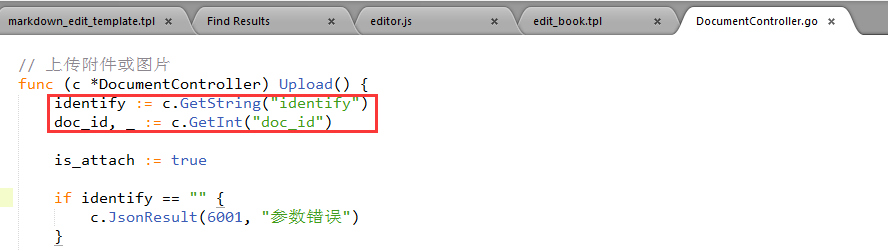




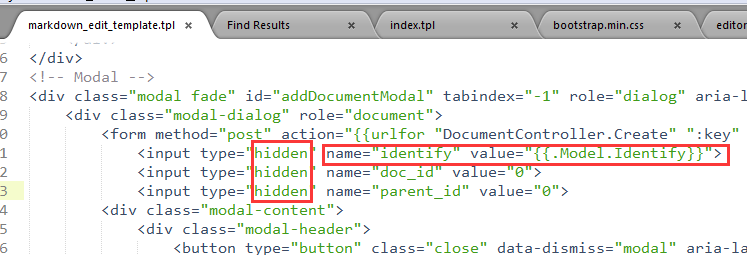


### Tpl—>Controller （webloader formdata）



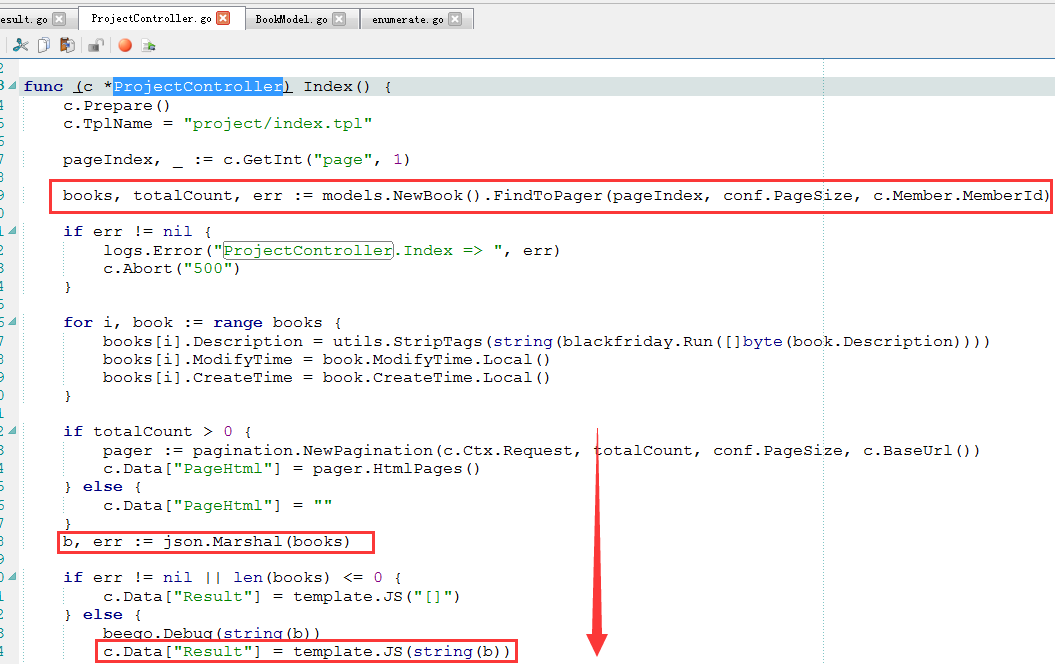


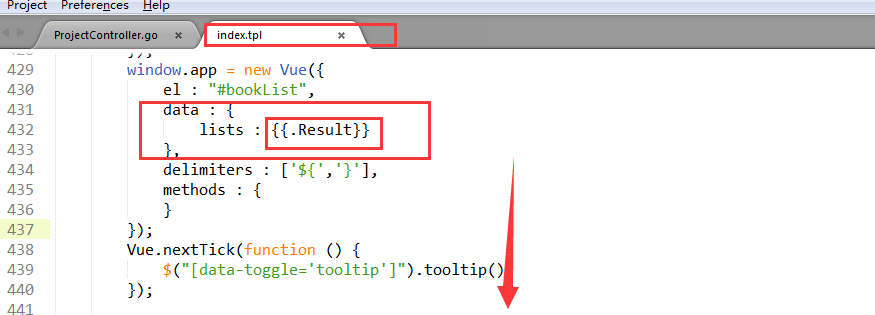
### Tpl—>Controller （通过隐藏控件->获取控件名字）

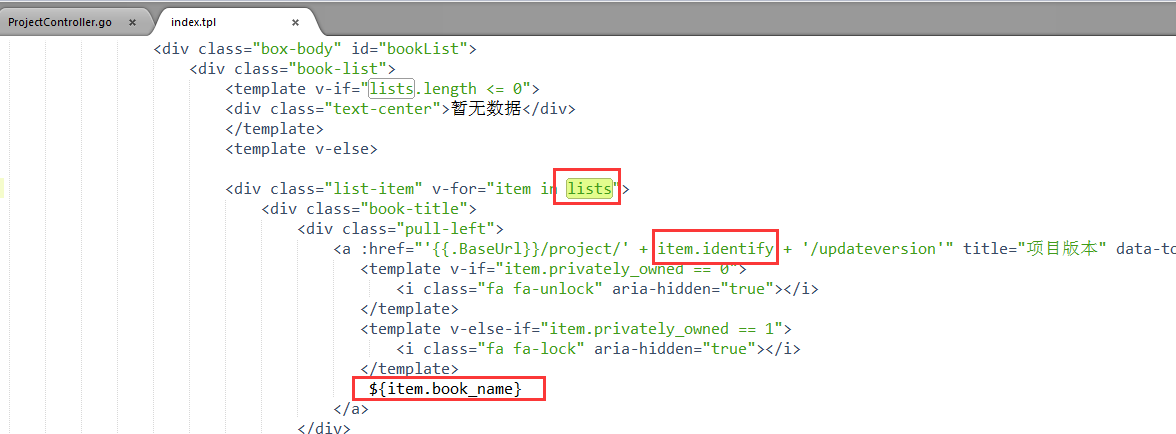




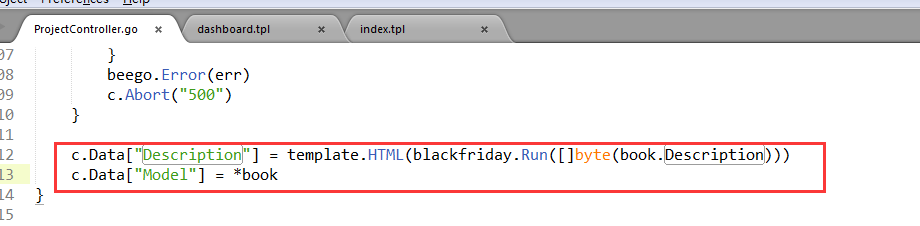
### Controlle—》Tpl （1, 通过给c.Data赋值 2 ajax post等待Controller返回json）







===================================~==================================





~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~





### Controlle—》Tpl (c.Ctx.Output.JSON)



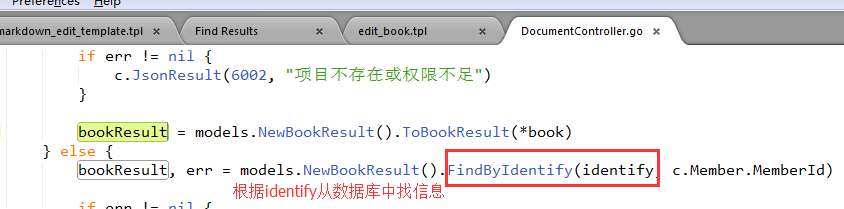


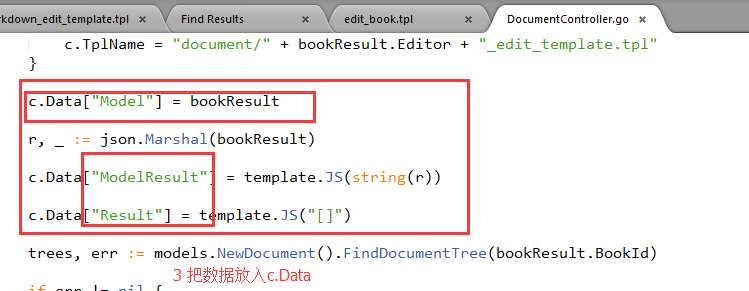
### 接上Post返回



## 上传前期处理







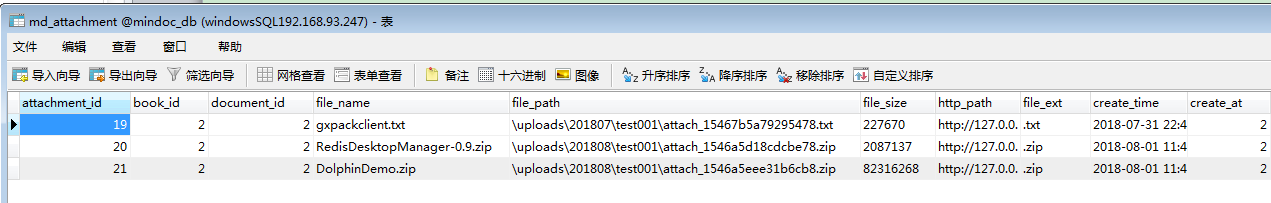


.Model.Identify

window.selectNode.id

## 上传后期处理

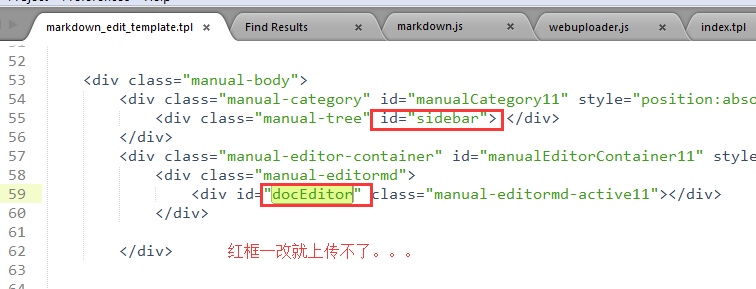
上传后：



保存后:

仅仅是关闭界面

## 点上传按钮没反应





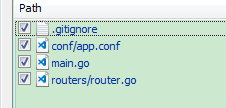


A: JS报错了，压根没概念。。。

## 版本列表

# AssetExpress

## 提交gitlab备注



D:\Kongfu\server\

F:\gowebserver

F:\beegotest

F:\zzs

F:\gowebserverminidoc



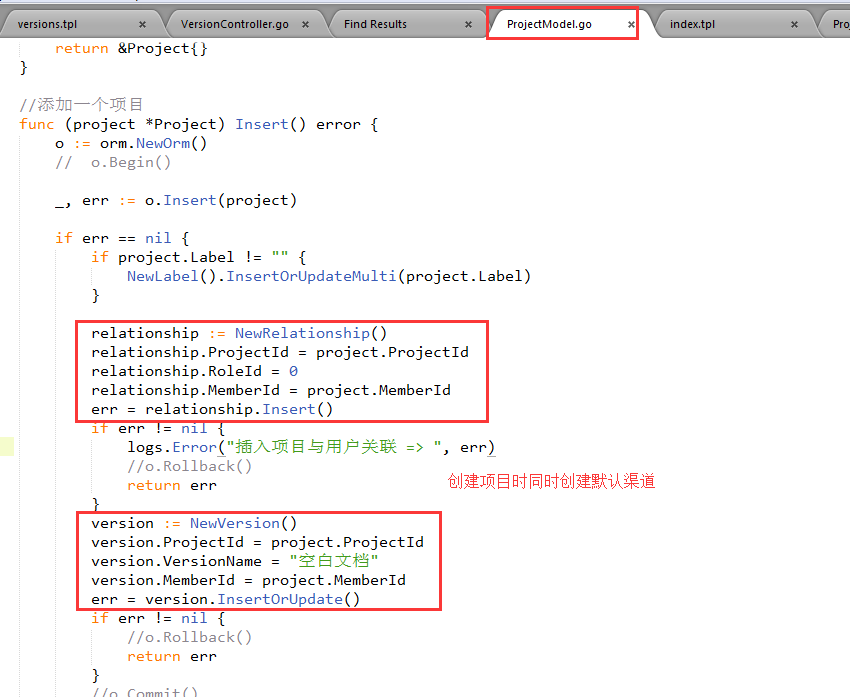
## 创建项目

### 数据传递 (ajax)

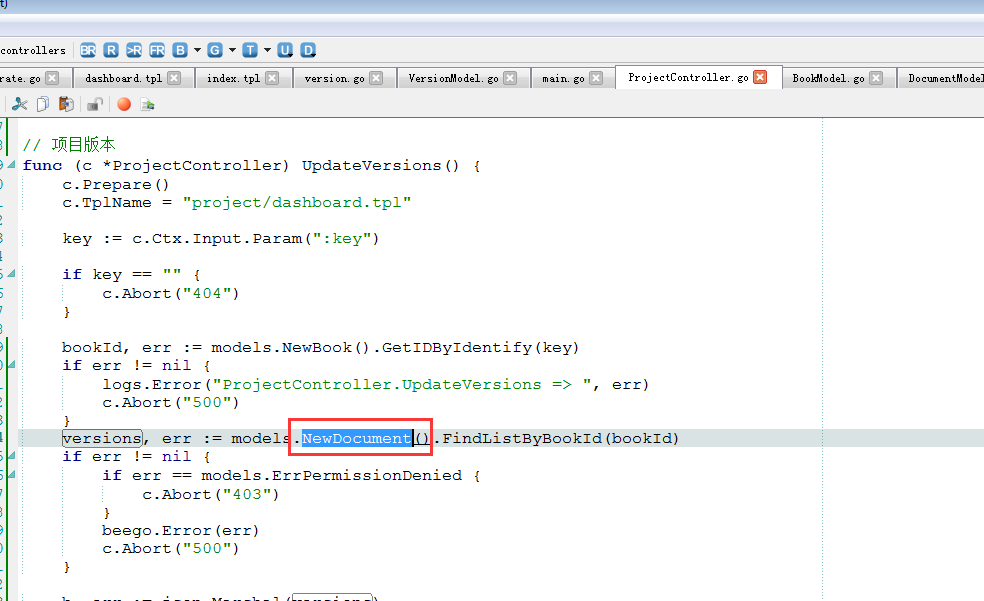




### 创建项目时同时创建默认渠道

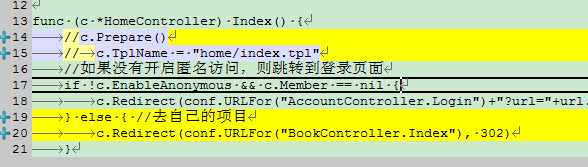


## 版本列表



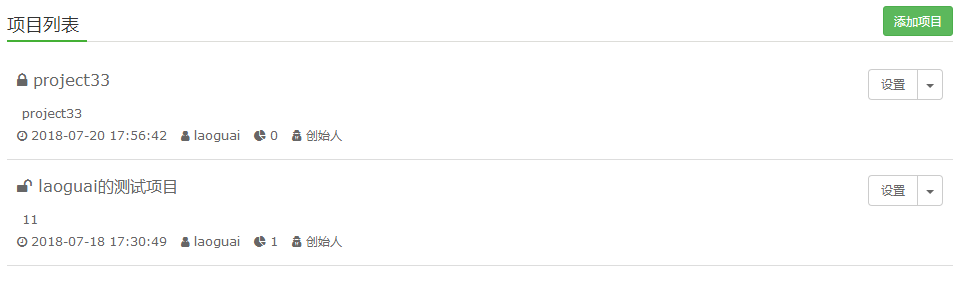


## 主页自动跳到/project

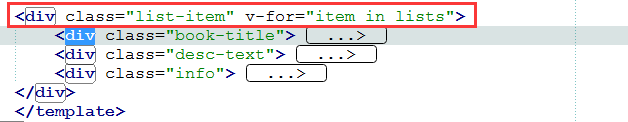


## 注册、登录

## 项目列表



循环:



## 上传、下载

### 路径

http://115.159.98.100:8990/1254459133/74122/cures/2.0.0.1/1254459133\_74122\_2.0.0.1\_20180712233401\_1120766625\_cures.ifs

Add a function for large files upload?

https://github.com/kataras/iris/issues/690

iris官方：

<https://github.com/kataras/iris/blob/master/_examples/http_request/upload-files/main.go#L38>

<https://stackoverflow.com/questions/36254522/golang-which-function-for-uploading-large-files>

<https://github.com/kataras/iris/issues/619>

<div class="modal fade" id="addBookDialogModal" tabindex="-1" role="dialog" aria-labelledby="addBookDialogModalLabel">

<div class="modal-dialog modal-lg" role="document" style="max-width: 650px; min-width: 400px;">

上传下载：

## Q&A

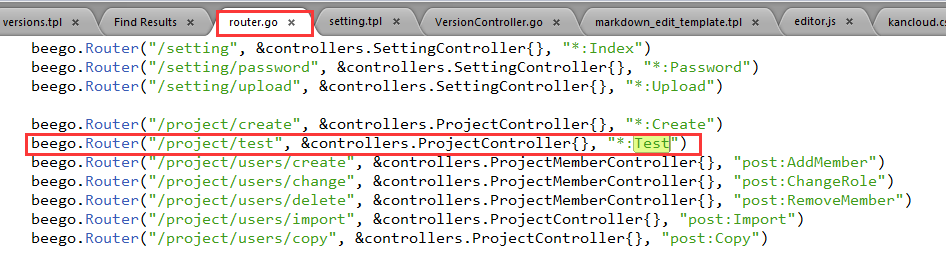
### 随机名字

fileName := "attach\_" + strconv.FormatInt(time.Now().UnixNano(), 16)

### Ajax调用跳不到go函数里去



竟然要加路由(如下)。。。



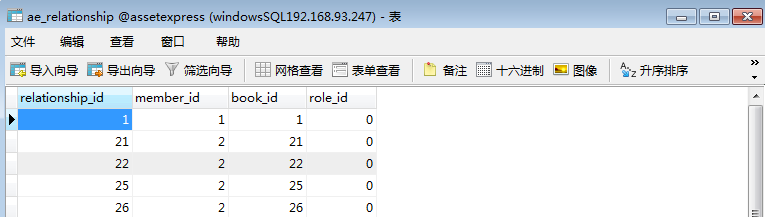


### 删除附件

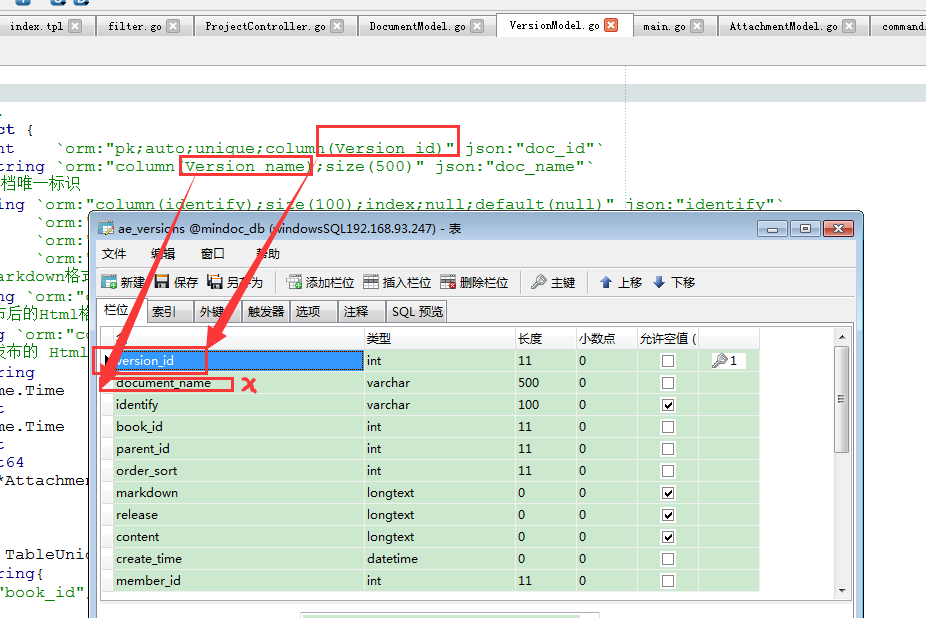
### 项目列表的横线不见了。。。



### 添加ProjectModel处理

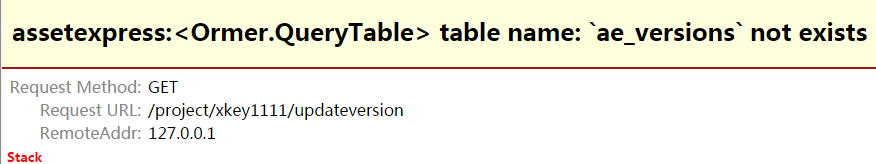


### 表字段对应关系



后面的json: xxx 是json序列化后对应的字段

### 添加的表提示未找到表

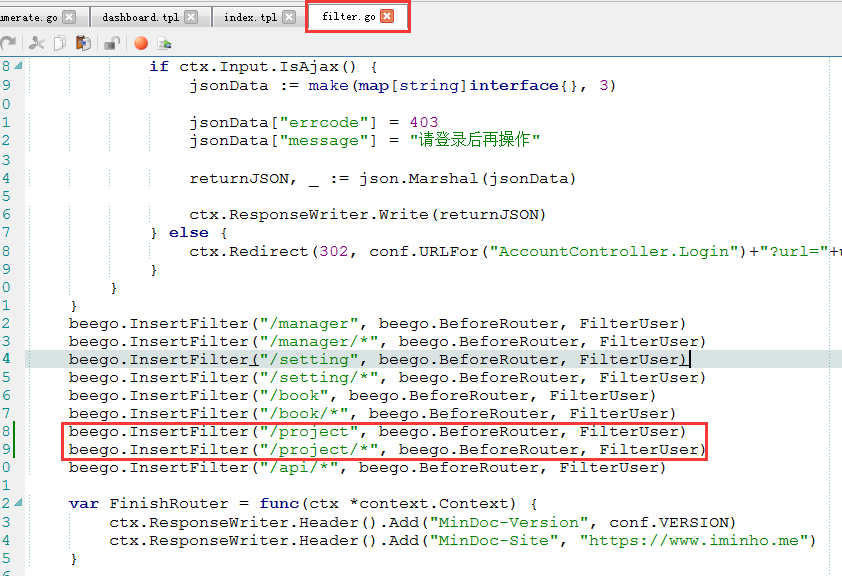


Model需要注册:



### 未登录情况下访问/project报错

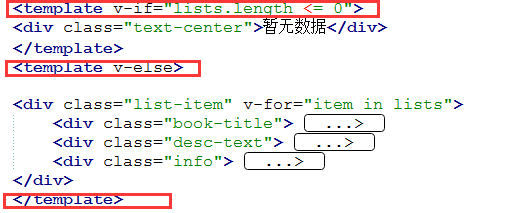
在filter.go中添加如下路径就可以了



# Web

## 语法

### v-if v-else



### v-for

# IRIS

## API

### 静态网页

app.StaticWeb("/public", "./web/public")

### 出错页面

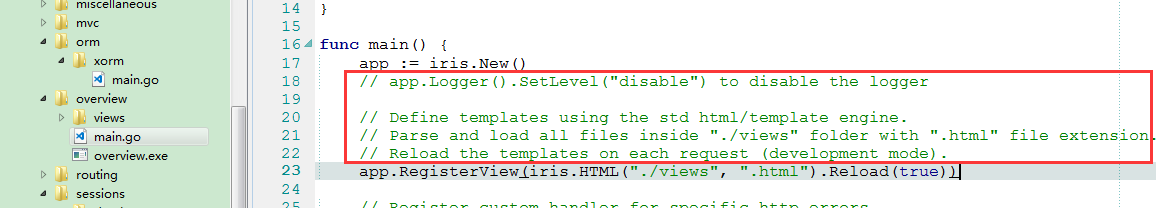
app.OnAnyErrorCode(**func**(ctx iris.Context) {

ctx.ViewData("Message", "啦啦到时")

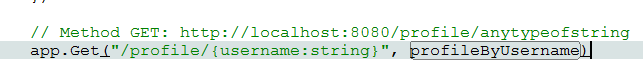
ctx.View("shared/error.html")

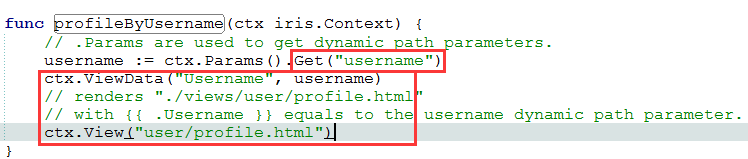
})

### 加载静态页面



### 路由路径带参数





### ctx.HTML

# Beego

## Go语法

### 循环遍历

for i, book := range books {

books[i].Description = utils.StripTags(string(blackfriday.Run([]byte(book.Description))))

books[i].ModifyTime = book.ModifyTime.Local()

books[i].CreateTime = book.CreateTime.Local()

}

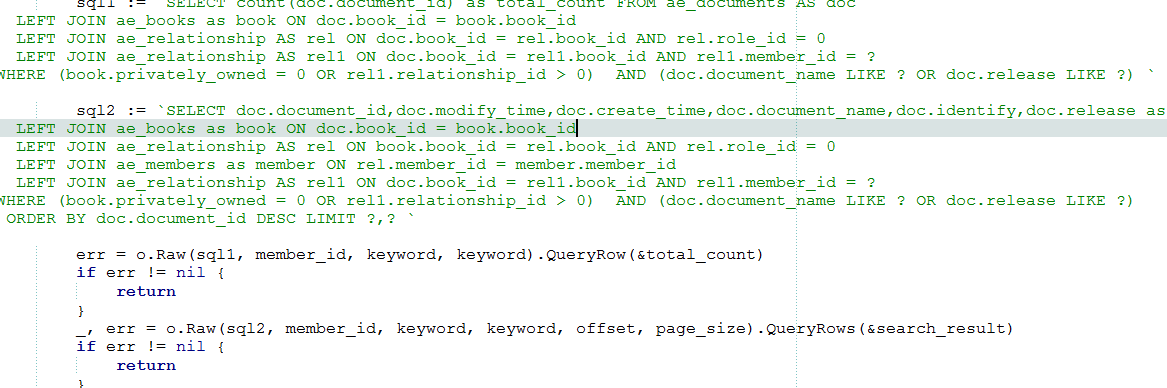
## 数据库查询

### o.Raw

o := orm.NewOrm()

err = o.Raw(sql1, memberId).QueryRow(&totalCount)

\_, err = o.Raw(sql2, memberId).QueryRows(&books)

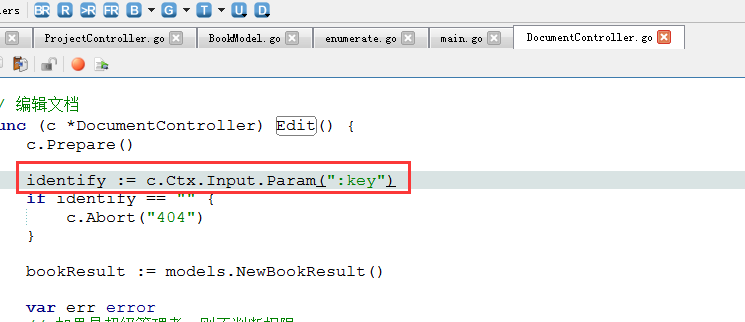


### o.QueryTable

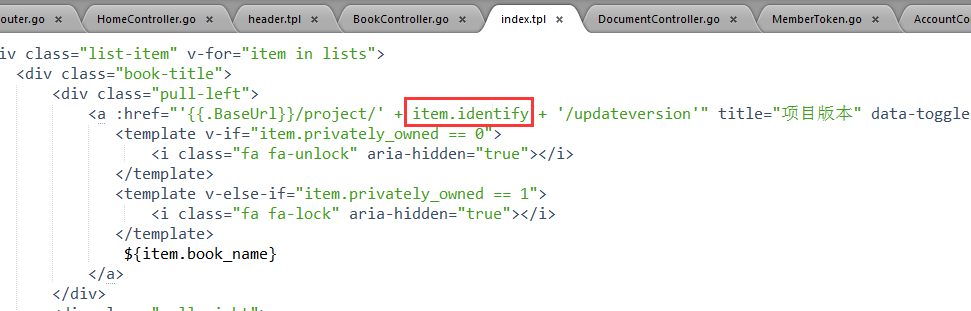
err := o.QueryTable(doc.TableNameWithPrefix()).Filter("book\_id", book.BookId).OrderBy("modify\_time").One(doc)

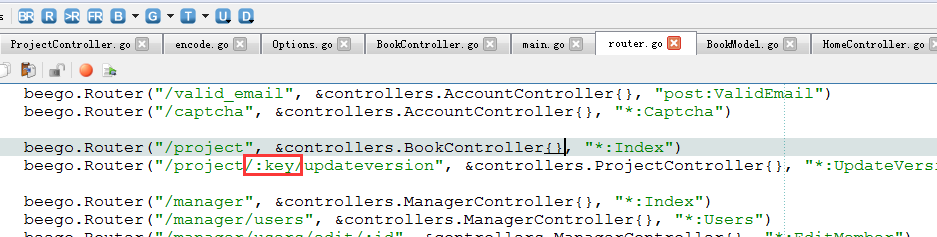
## 数据传递

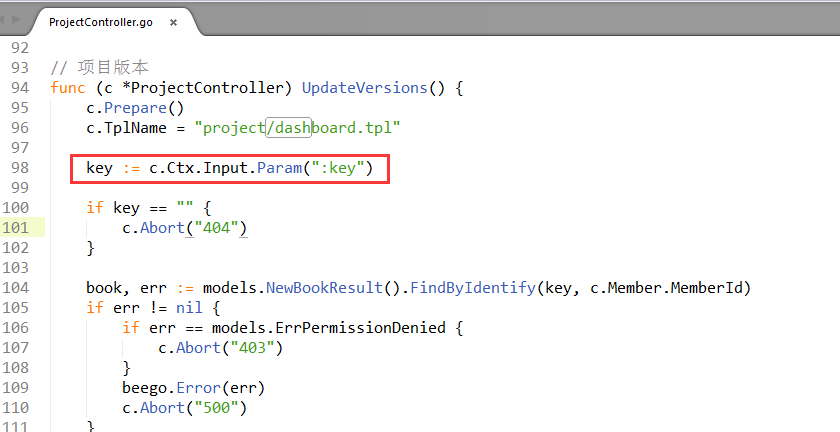
### 解析路径上的参数



### Tpl—>Controller

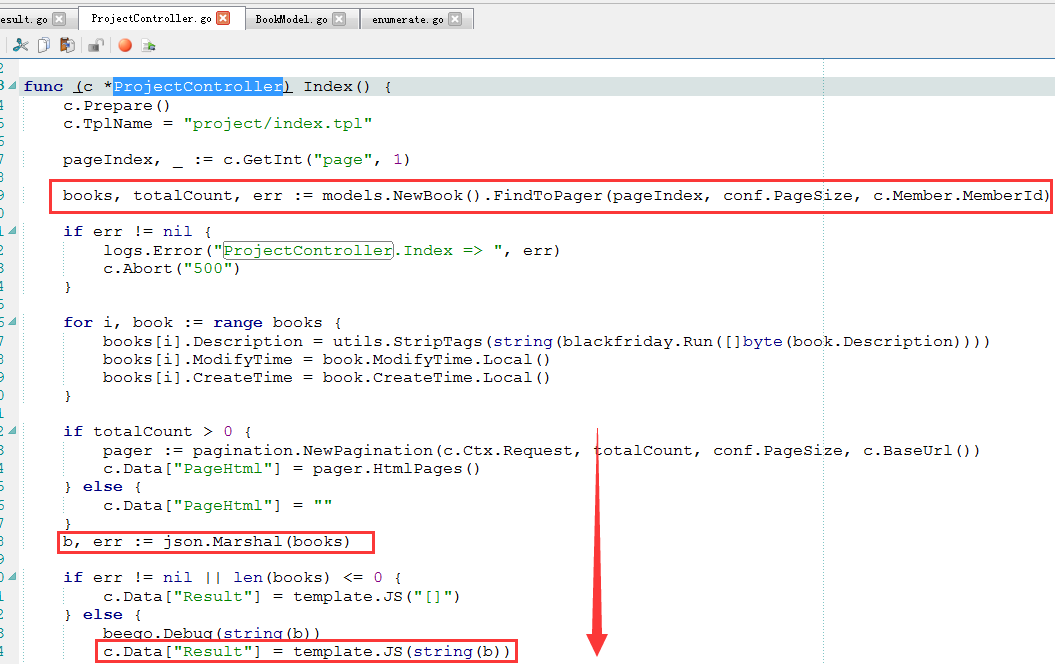


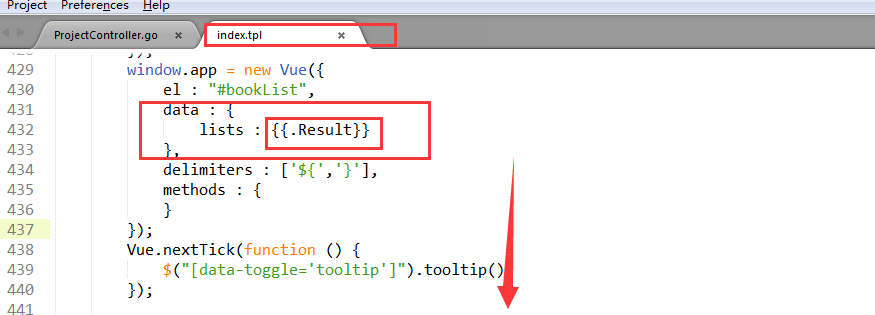


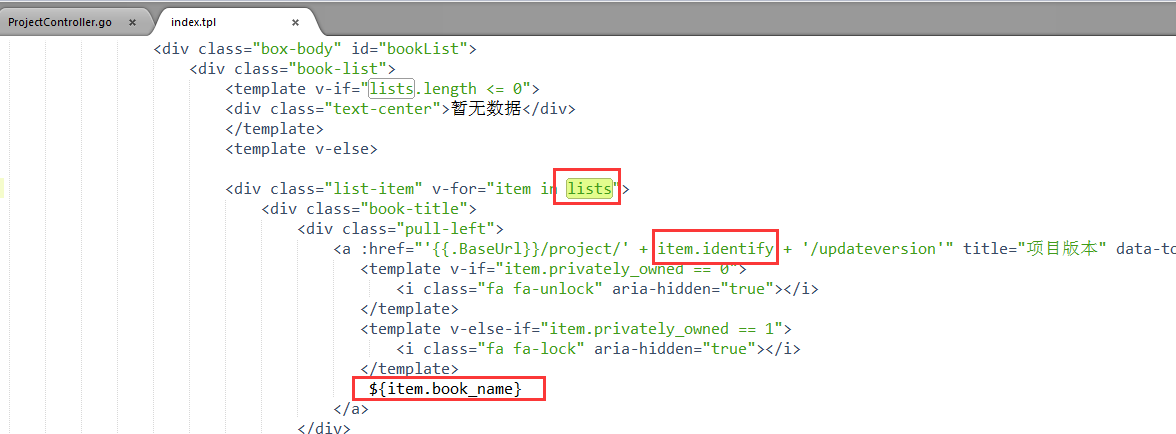


<a href="{{urlfor "DocumentController.Edit" ":key" .Model.Identify ":id" ""}}" class="btn btn-default btn-sm pull-right" target="\_blank"><i class="fa fa-edit" aria-hidden="true"></i> 编辑</a>

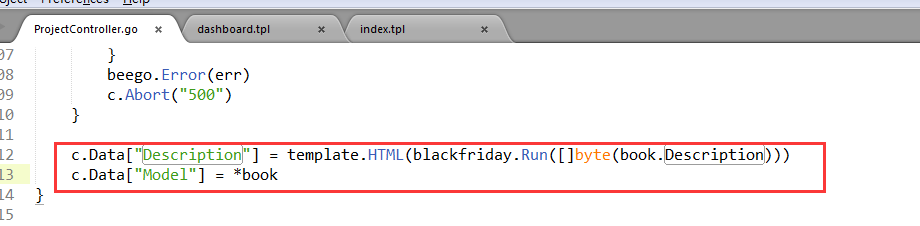
### Controlle—》Tpl （1, 通过给c.Data赋值 2 ajax post等待Controller返回json）







===================================~==================================





~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~





### 接上Post返回



## Basic

### 日志

beego.Trace("11111111testing")

beego.Debug("2222")

beego.Error("HeHe")

beego.Warn("warning")

### bee命令

创建基础框架

bee new my-web-app

打包执行性程序

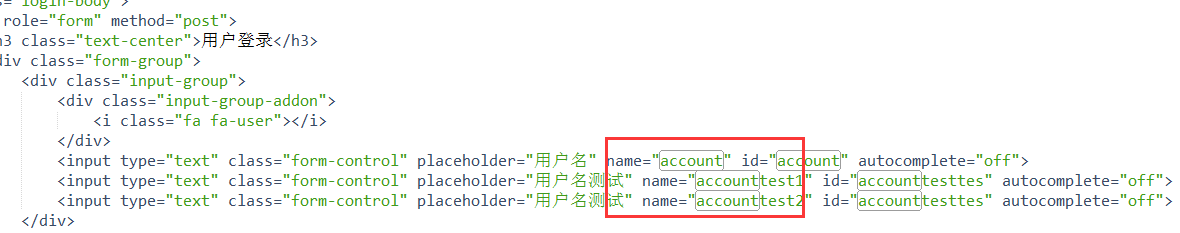
bee pack

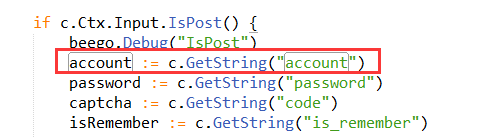
api框架？

bee api my-api

bee generate controller hello

### 获取post数据:





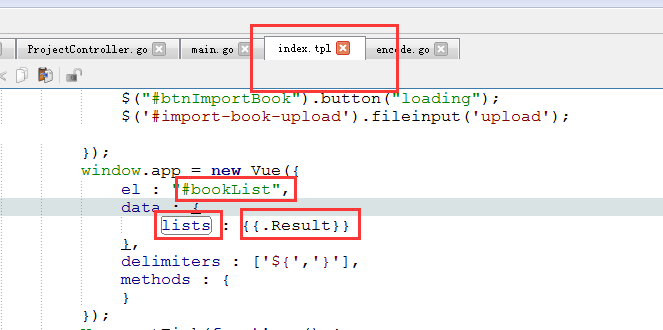
### 登录跳转的页面在这里。。：

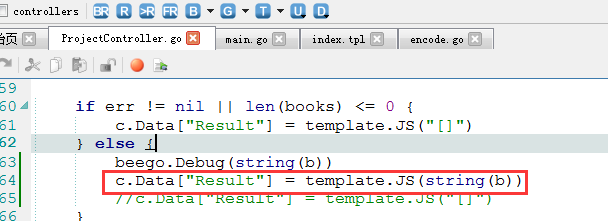


### Tpl网页的数据源从哪来 (比如项目列表)

Go（c.data）🡪vue🡪tpl

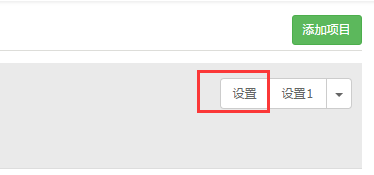






## UI

### 按钮链接？



<a :href="'{{.BaseUrl}}/project/' + item.identify + '/dashboard'" class="btn btn-default">设置1</a>



<a href="{{urlfor "DocumentController.Edit" ":key" .Model.Identify ":id" ""}}" class="btn btn-default btn-sm pull-right" target="\_blank"><i class="fa fa-edit" aria-hidden="true"></i> 编辑</a>



### 箭头



### 提示





### 跳转



## FrameWork

### Conf

### Controllers

### Models

### Routers

### Static

css

image

js

### Views

.tpl文件

## 第三方应用

### MinDoc

**依赖：**

F:\gowebserver>go get github.com/garyburd/redigo/redis

F:\gowebserver>go get github.com/bradfitz/gomemcache/memcache

F:\gowebserver>go get github.com/go-sql-driver/mysql

F:\gowebserver>go get github.com/kardianos/service

**关于golang.org/x包问题**

git clone https://github.com/golang/net.git $GOPATH/src/github.com/golang/net

git clone https://github.com/golang/sys.git $GOPATH/src/github.com/golang/sys

git clone https://github.com/golang/tools.git $GOPATH/src/github.com/golang/tools

git clone https://github.com/golang/image.git F:\gowebserver\src\golang\image

ln -s $GOPATH/src/github.com/golang $GOPATH/src/golang.org/x

- beego 1.8.1

- mysql 5.6

- editor.md

- bootstrap 3.2

- jquery 库

- layer 弹出层框架

- webuploader 文件上传框架

- Nprogress 库

- jstree 树状结构库

- font awesome 字体库

- cropper 图片剪裁库

- layer 弹出层框架

- highlight 代码高亮库

- to-markdown HTML转Markdown库

- wangEditor 富文本编辑器

**#** **主要功能**

- 项目管理，可以对项目进行编辑更改，成员添加等。

- 文档管理，添加和删除文档等。

- 评论管理，可以管理文档评论和自己发布的评论。

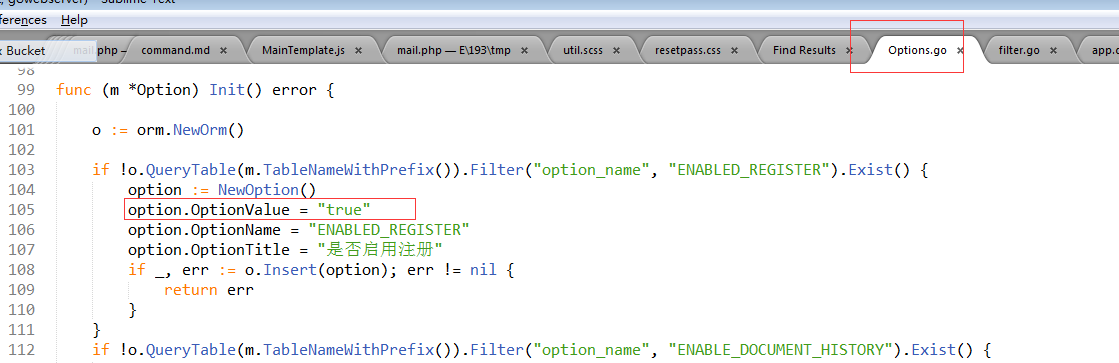
- 用户管理，添加和禁用用户，个人资料更改等。

- 用户权限管理 ， 实现用户角色的变更。

- 项目加密，可以设置项目公开状态，私有项目需要通过Token访问。

- 站点配置，可开启匿名访问、验证码等。

是否启用注册：



数据源来自数据库

