链滴

Goreleaser + TraivsCI 发布 gopo 项目

作者：FengY95
原文链接：https://ld246.com/article/1614736458729
来源网站：链滴

Goreleaser和TraivsCI 这边就不再过多介绍，可以去它们官网上了解。这边主要是介绍gopo以及如把它发布出去供人使用。那这篇文章也相应的分为两个部分。

gopo

gopo 说白了就是解析go中的结构体，生成对应的结构体字段常量表。也就是说：

```
type SomeTable struct{
        Id       int     `gorm:"primary_key" json:"id"` //记录id
        Height   float32 `json:"height"`               //身高
        Weight   float32 `json:"weight"`               //体重
}
```

解析以上结构体后，生成如下代码：

```
// 自动生成
const (
    SomeTableTableName = "cms.some_table"

    SomeTableId = "id"
    SomeTableHeight = "height"
    SomeTableWeight = "weight"
)
```

可以看到，生成了这个结构体中每个字段对应的常量值，这样有什么作用呢？主要是在一些ORM框中，虽然杜绝了直接写SQL，但也会有写死字段名的情况，如下例所示：

```
func FindHeight(id int){
    // 只取这两个字段
    db.Select("id","height").Find(&SomeTable{},id)
}
func FindWeight(id int){
    db.Select("id","weight").Find(&SomeTable{},id)
}
// 这种通用的Select就更难管理了
func FindBy(id int,selects []string){
    db.Select(selects).Find(&SomeTable{},id)
}
```

可以看到，假如我只想去两个字段，还得把字段名写死在代码中。如果是大型的项目，这就有些不好理了，万一出现一些改名、删字段的极端情况，还得去代码中搜索，也容易遗漏。所以我们最好能把个数据库字段名提取出来作为常量，利用IDE管理起来就很方便了。

```
a := FindBy(1,[]string{SomeTableId,SomeTableHeight})
```

这样，即保证了灵活性，也不担心项目里到处都是写死的数据库字段名。那么对于表多、字段多的情，使用gopo自动生成就很方便了。

gopo的原理也很简单，就是解析go的AST语法树，然后生成对应的代码，实现的思路和代码都借鉴了waggo/swag，这是一个很牛逼的项目，在此表示感谢。

gopo的使用也很简单，提供了很多可选参数，最简单用法就是：

```
$ gopo
```

那么它就会自动在当前目录下寻找go文件，收集里面的所有结构体，并为每个结构体生成对应的常量。更多用法在项目的README.md中有说明，也可以通过--help去查看此命令的帮助文档。

[Goreleaser&TraivsCI]

进入本文的压轴，其实对go语言来说，还有一块很适合的应用就是编写一些命令行程序，简直不要太便。代码编写完成后，打包直接能生成Liunx\Mac\Windows三大平台的可执行文件，也不需要考虑装环境、语言版本等问题，比Python、Java什么的方便太多。当然这里不是想挑起语言之争，只是说在打包、编译方面，go做的比较好。

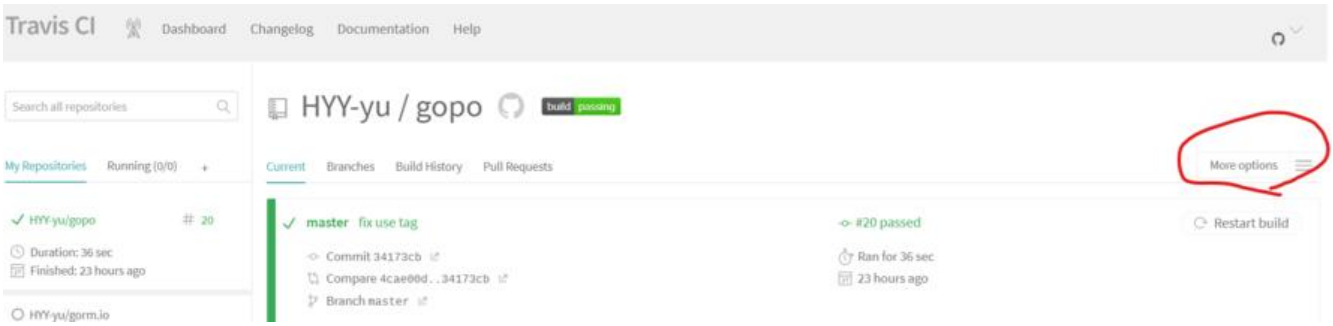
那么如果加上goreleaser和CI，代码一推上去直接打包完毕同时更新到homebrew、scoop等包管理件上，就更加的便捷了。这个项目的.goreleaser.yml和.traivs.yml这两个配置，基本上其他的go语言写的命令行程序也可以直接拿来使用，有需要者自取即可。

简单介绍一下这整个流程：

1. TraivsCI 检测到代码提交，自动运行打包任务
2. 打包任务调用goreleaser进行打包
3. 通过 .goreleaser.yml中的配置，goreleaser命令会自动打包并发到github的release模块，并且送到homebrew或者scoop等包管理工具上。

这里要注意的就是两点：

1. TravisCI中需要设置github的personal access token，因为goreleaser需要往github上推包。



在图片上红圈标注的位置点击Setting，然后在Environment Variables中加一项GITHUB_TOKEN即。

2. .goreleaser.yml会自动建立homebrew、scoop的repo，如果直接拿过来是不行的（这是我的账下的配置），需要一定的修改：

```
before:
  hooks:
    # You may remove this if you don't use go modules.
    - go mod download
builds:
  - env:
      - CGO_ENABLED=0
    main: cmd/main.go
    goos:
      - linux
      - darwin
      - windows
```

原文链接：Goreleaser + TraivsCI 发布 gopo 项目

```
    goarch:
      - 386
      - amd64
archives:
  - name_template: '{{ .ProjectName }}_{{ .Os }}_{{ .Arch }}{{ if .Arm }}v{{ .Arm }}{{ end }}'
    replacements:
      darwin: Darwin
      linux: Linux
      windows: Windows
      386: i386
      amd64: x86_64
checksum:
  name_template: 'checksums.txt'
snapshot:
  name_template: "{{ .Tag }}-next"
changelog:
  sort: asc
  filters:
    exclude:
      - '^docs:'
      - '^test:'
brews:
  -
    # 项目名
    name: gopo

    # Github repository 需要改成自己的
    tap:
      owner: HYY-yu
      name: homebrew-tap

    # 需要改成自己的
    url_template: "https://github.com/HYY-yu/gopo/releases/download/{{ .Tag }}/{{ .ArtifactNa
e }}"

    # Git author used to commit to the repository.
    # Defaults are shown.
    commit_author:
      name: goreleaserbot
      email: goreleaser@carlosbecker.com

    # Folder inside the repository to put the formula.
    # Default is the root folder.
    folder: Formula

    # Caveats for the user of your binary.
    # Default is empty.
    caveats: "gopo -v"

    # Your app's homepage.
    # Default is empty.
    # homepage: "https://example.com/"

    # Your app's description.
```

```
    # Default is empty.
    description: "Gopo"

    # Specify for packages that run as a service.
    # Default is empty.
    plist: |
      <?xml version="1.0" encoding="UTF-8"?>
      ...

    # Custom install script for brew.
    # Default is 'bin.install "program"'.
    install: |
      system "rm -f #{bin}/gopo"
      bin.install "gopo"
scoop:
  # 需要改成自己的
  url_template: "https://github.com/HYY-yu/gopo/releases/download/{{ .Tag }}/{{ .ArtifactNam
 }}"

  # Repository to push the app manifest to.
  bucket:
    owner: HYY-yu
    name: scoop-bucket

  # Git author used to commit to the repository.
  # Defaults are shown.
  commit_author:
    name: goreleaserbot
    email: goreleaser@carlosbecker.com

  # The project name and current git tag are used in the format string.
  commit_msg_template: "Scoop update for {{ .ProjectName }} version {{ .Tag }}"

  # Persist data between application updates
  persist:
    - "data"
    - "config.toml"
```

总结

那么Goreleaser+TraivsCI整个流程基本也介绍完毕了，现在只需要为master分支打一个tag，比如v0.
.0，那么直接就会生成对应版本的可执行文件，并发布出去。

使用者利用brew upgrade gopo 更新即可。