go mod proxy

发布日期	版本号	作者	描述
2020年10月04日	0.9.2	zwb	

目录

更新日志

更新日期	版本号	作者	描述
2020年10月04日	0.9.2	zwb	实现代理服务，并增加文件上传

关于勘误

由于时间水平都比较有限，所有文档中难免会出现些纰漏和错误，如果大家发现了一些问题，欢迎大家提交issue反馈

Go包管理

在 `Go1.5` 之前用 `GOPATH` 以及 `GOROOT` 这两个环境变量来管理包的位置， `GOROOT` 为 `Go` 的安装目录，以及编译过程中使用到的系统库存放位置，如fmt。 `Go1.5` 到 `Go1.7` 开始稳定到 `Vendor` 方式，即依赖包需要放到 `$GOPATH/src/vendor` 目录下，这样每个项目都有自己的 `vendor` 目录，但是如果依赖同样的三方包，很容易造成资源重复，Go vendor 出现了几种主流的管理工具，包括 `godep`、`govendor`、`golide` 等。

在 `Go1.11` 之前， `GOPATH` 是开发时的工作目录，其中包含三个子目录：

- src目录：存放go项目源码和依赖源码，包括使用 `go get` 下载的包
- bin目录：通过使用 `go install` 命令将 `go build` 编译出的二进制可执行文件存放于此
- pkg目录：go源码包编译生成的lib文件存储的地方

在 `Go1.11` 之前， `import` 包时的搜索路径

- `GOROOT/src`: 该目录保存了Go标准库代码(首先搜寻导入包的地方)
- `GOPATH/src`: 该目录保存了应用自身的各个包代码和第三方依赖的代码
- `./vendor` ： vendor 方式第三方依赖包（如果支持Vendor）

在 Unix 和类 Unix 系统上，`GOPATH` 默认值是 `$HOME/go`，Go1.11 版本后，开启 GO Modules 后，`GOPATH` 的作用仅仅为`存放`依赖的目录了。

在 `Go` 的 `1.11` 版本之前，`GOPATH` 是必需的，且所有的 `Go` 项目代码都要保存在 `GOPATH/src` 目录下，也就是如果想引用本地的包，你需要将包放在 `$GOPATH/src` 目录下才能找得到。`Go` 的 `1.11` 版本之后，`GO` 官方引入了 `Go Modules`，不仅仅方便的使用我们的依赖，而且还对依赖的版本进行了管理。

在Go1.11后通过 `go mod vendor` 和 `-mod=vendor` 来实现 Vendor 管理依赖方式。本来在 `vgo` 项目(Go Modules前身)是要完全放弃 `vendor`，但是在社区反馈下还是保留了。总之就是在 Go.1.11 之后需要开启 Go Modules 条件下才能使用 Vendor，具体地感兴趣或还沿用了 Vendor 的朋友可以去了解下，不过建议以后仅使用 Go Modules 包管理方式了

总体介绍

虽然`Go1.11` 引入了Go Modules的包管理机制，但是依然存在一些问题：

1. 项目需要托管在公共仓库上，比如GitHub，GitLab等，对于内部私有项目存在一些问题
2. 虽然一些办法解决私有项目访问权限的问题，但是配置过于繁琐，详见 Go Modules私有项目配置

为了解决以上问题，开发了`go-mod-proxy`，是一个go代码服务，也是一个私有管理服务，支持内网运行。

功能列表

- [x] 代理

  只需要通过设置`GOPROXY`环境变量就可以实现代理，比如：

  ```
  export GOPROXY=https://mirrors.aliyun.com/goproxy/,https://goproxy.cn,direct
  ```

- [x] 私有项目批量发布

  只需要将本地`$GOPATH/pkg/mod/cache/download`批量拷贝至`go-mod-proxy`中的仓库目录即可

- [x] 私有项目API发布

  提供HTTP API接口，发布mod至`go-mod-proxy`仓库中

安装步骤

方式一：安装最新版本

git clone源码，然后执行`scripts`中的`build.sh`脚本进行打包，生成文件在项目的`releases`目录下

方式二：直接Releases中直接下载对应版本

windows版本： go-mod-proxy_0.9.2_windows_amd64.exe

linux版本：go-mod-proxy_0.9.2_linux_amd64

mac版本：go-mod-proxy_0.9.2_darwin_amd64

下载完成后，重名为简单名字比如`go-mod-proxy.exe`，然后配置好PATH环境变量就可以直接使用了

启动服务

启动参数说明

```
go-mod-proxy.exe --help
使用说明:

Usage:
  go-mod-proxy.exe

Flags:
      --apiport int          API端口，默认端口为代理端口+1，用于上传私有包
  -h, --help                 帮助信息
      --host string          绑定的host
  -p, --port int             代理端口 (default 8081)
  -r, --repository string    本地仓库目录 (default "./data")
  -v, --version              版本信息
```

示例:

输入:

```
go-mod-proxy.exe -p 8081 -r repo
```

输出:

```
go-mod-proxy.exe -p 8081 -r repo
2020/10/04 19:02:45 本地仓库目录: G:\go_workspace\go-mod-
proxy\releases\0.9.2\windows\repo
2020/10/04 19:02:45 启动API服务 ,监听地址[:8082]
2020/10/04 19:02:45 启动代理服务,监听地址[:8081]
```

注意实现

1. 私有项目代理注意

2）配置GONOSUMDB

`GONOSUMDB` Go环境变量用于设置哪些前缀的模块都被视为私有模块，不进行HASH检验，通过逗号分隔配置多个匹配路径，因为的私有项目未至公网，SUMDB无法收集到校验信息，导致从`go-mod-proxy`下载的私有项目包错，所以客户端（开发者）需要配置校验例外，比如：

```
go env -w GONOSUMDB=github.com/wenit
```

也可通过关闭整个HASH校验，不建议这么做

```
GOSUMDB="off"                                                    # 关闭校验，任何模块可以被使用
```