# 概述

## module

- 一个Module--多个package; 一个package--多个.go文件.

- 声明包名，声明后其他包可以引用

- go mod init: 初始化module , 项目目前下生成go.mod

-- 有go.mod目录为可以供其他调研得

· go mod init 创建了一个新的module, 并在根目录创建一个go.mod文件去描述该module(如module path, 依赖项(指定版本module)).

· go build, go test, 以及其他的编译命令会下载需要的依赖, 并把依赖项添加到go.mod文件中.

· go list -m all打印当前module, 和所有的依赖项(直接/间接依赖)

· go get 获取/更新一个依赖项.

· go mod tidy 清理无用的依赖项.

-- 设置 ：

---- 打卡 go module 模式：依赖解决方案

在 Go modules 中，我们能够使用如下命令进行操作：

| 命令 | 作用 |
| --- | --- |
| go mod init | 生成 go.mod 文件 |
| go mod download | 下载 go.mod 文件中指明的所有依赖 |
| go mod tidy | 整理现有的依赖 |
| go mod graph | 查看现有的依赖结构 |
| go mod edit | 编辑 go.mod 文件 |
| go mod vendor | 导出项目所有的依赖到vendor目录 |
| go mod verify | 校验一个模块是否被篡改过 |
| go mod why | 查看为什么需要依赖某模块 |

--- go env 环境变量：

---- GO111MODULE：推荐on，表示启用 Go modules，

---- GOPROXY：设置代理，加快下载速度

$ go env -w GOPROXY=https://goproxy.cn,direct，

- 获取本地包

replace example.com => ../example2.com //go.mod文件

。。

Import example.com

## package

- package 和 package所在的目录名, 一般是一致的. 当然也可以不一致

- 使用其他包： import 别名 “包文件的路径”

- 引入的包位置：

-- 标准库包: 源码在$GOROOT/src下

-- 第三方依赖包: 源码在$GOPATH/src下

-- 本地包：import module路径/包名称

---    import ( test "test.com/m/test")

## 常用命令

= 是赋值， := 是声明变量并赋值。

## 测试

- go test

-- 执行\_test.go文件中的以Test开头命名的函数

-- 在包含\_test.go的文件夹目录下运行

## 部署

- go build

// go build 会生成可实行文件放在当前目录中。

//go install 则会把它放到 $GOPATH/bin 中。

## workspace

- go work init [path]

// 生成go.work文件，里面定义版本，空间的相对路径，加入[path]下的modules

// 在go.work路径下运行go run [path]时，根据[path]寻找modules

- go work use [path] 加入modules路径

## Go and Gin web service

- 路由：router := gin.Default()

Router.GET([url],[func name])

router.GET("/albums", getAlbums)

router.GET("/albums/:id", getAlbumByID)

router.POST("/albums", postAlbums)

- 路由中包含参数:id