**Golang学习笔记**

# 1 golang安装

## 1.1 window下安装

## 1.2 linux下安装

# 2 第三方依赖包管理

一般项目放在源码$GOPATH/src，依赖包放在$GOPATH/pkg，编译生成执行文件放在$GOPATH/bin。上面的路径都是固定的，第三方依赖包越来越多，而且存放在同一个目录下，移植项目到其他地方时又要重新下载依赖包。

## 2.1 godep依赖包管理

**(1) 安装**

**go get github.com/tools/godep**

安装完后把godep可执行文件复制到$GOPATH/bin目录下

**(2) godep命令**

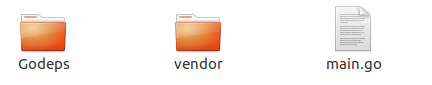
|  |  |
| --- | --- |
| 命令 | 功能 |
| save | 复制项目中的依赖文件到vendor，生成列表文件到Godeps |
| go | 运行go工具原生命令 |
| get | 下载和安装使用指定的依赖关系的软件包到$GOPATH/src |
| path | 打印GOPATH依赖代码 |
| restore | 检查列出GOPATH的依赖版本 |
| update | 选择更新软件包或go版本 |
| diff | 显示当前和以前保存的依赖集之间的差异 |

**(3) 复制依赖文件到项目下**

对于没有使用过godep工具的项目，在项目根目录下执行命令：

**godep save**

在项目下生成Godeps和vendor文件夹。Godeps存放依赖列表和版本的json文件，vendor存放依赖源码文件，如下图所示，最后使用git提交项目文件和依赖文件到github上即可。



注：godep项目必须在$GOPATH/src操作

## 2.2 gb依赖包管理

使用gb依赖包管理可以很方便独立管理各个项目的依赖包，每个项目下都有独立的四个文件夹bin、pkg、src、vendor，直接把这四个文件夹提交到github即可，以后clone项目时一并把依赖包也clone下来，不必另外去找依赖包。而且项目文件夹可以在任何地方创建、编译和运行，不必固定在$GOPATH路径下。

**(1) 安装gb**

**go get github.com/constabulary/gb/...**

安装完后，在$GOPATH/bin(或$GOPATH/bin:)下生成gb和gb-vendor两个可执行文件，最后把两个执行文件放入你的$GOPATH/bin目录中即可。

**(2) gb和gb-vendor命令**

如果$GOPATH/bin目录下已经有gb和gb-vendor执行文件，执行gb命令没找到，在linux下可能没有在~/.bash\_profile或~/.bashrc(ubuntu)添加export PATH=$PATH:$GOPATH/bin

下面是gb和gb-vendor命令列表

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | gb 命令列表 | | | 命令 | **功能** | | build | 编译包 | | doc | 显示文档 | | env | 打印项目的环境变量 | | generate | 处理源代码生成Go文件 | | info | 显示项目的信息 | | list | 显示项目下的所有包 | | test | 执行测试 | | |  |  | | --- | --- | | gb vendor 命令列表 | | | 参数 | **功能** | | fetch | 获取一个远程依赖 | | update | 更新一个本地依赖 | | list | 每行一个列出所有依赖 | | delete | 删除一个本地依赖 | | purge | 清除所有未引用的依赖 | | restore | 从manifest清单文件还原依赖 | |

**(3) 测试项目**

|  |  |
| --- | --- |
| 新建项目目录 | mkdir simpleJson |
| 进入目录 | cd simpleJson |
| 新建src/simpleJson目录 | mkdir -p src/simpleJson |
| 新建vendor/src目录 | mkdir -p vendor/src |
| 编辑main.go文件 | vim src/simpleJson/main.go |
| // main.go文件内容  **package** main  **import** **(**  "fmt"  **.** "github.com/bitly/go-simplejson"  **)**  **func** main**()** **{**  jsonData **:=** `{  "user": {  "name": "张三",  "age": 25,  "hobbies": ["篮球", "音乐", "旅游"]  }  }`  js**,** err **:=** NewJson**([]byte(**jsonData**))**  **if** err **!=** **nil** **{**  fmt**.**Println**(**err**.**Error**)**  **return**  **}**  name**,** **\_** **:=** js**.**Get**(**"user"**).**Get**(**"name"**).**String**()**  age**,** **\_** **:=** js**.**Get**(**"user"**).**Get**(**"age"**).**Int**()**  hobbies**,** **\_** **:=** js**.**Get**(**"user"**).**Get**(**"hobbies"**).**StringArray**()**  fmt**.**Println**(**name**,** age**,** hobbies**)**  **}**  //输出结果：张三 25 [篮球 音乐 旅游] | |
| 添加第三方依赖包 | gb vendor fetch "github.com/bitly/go-simplejson" |
| 编译和执行程序 | gb build simpleJson  ./bin/simpleJson-linux-amd64 |
| 查看项目目录树tree -a |  |