Golang 中的 go mod 以及 Golang 包详解

主讲教师：（大地）

合作网站：www.itying.com （IT 营）

我的专栏：https://www.itying.com/category-79-b0.html

一、Golang 中包的介绍和定义

包（package）是多个 Go 源码的集合，是一种高级的代码复用方案，Go 语言为我们提供了很多内置包，如 fmt、strconv、strings、sort、errors、time、encoding/json、os、io 等。

Golang 中的包可以分为三种：1、系统内置包 2、自定义包 3、第三方包

系统内置包: Golang 语言给我们提供的内置包，引入后可以直接使用，如 fmt、strconv、strings、sort、errors、time、encoding/json、os、io 等。

自定义包：开发者自己写的包

第三方包：属于自定义包的一种，需要下载安装到本地后才可以使用，如前面给大家介绍的"github.com/shopspring/decimal"包解决 float 精度丢失问题。

二、Golang 包管理工具 go mod

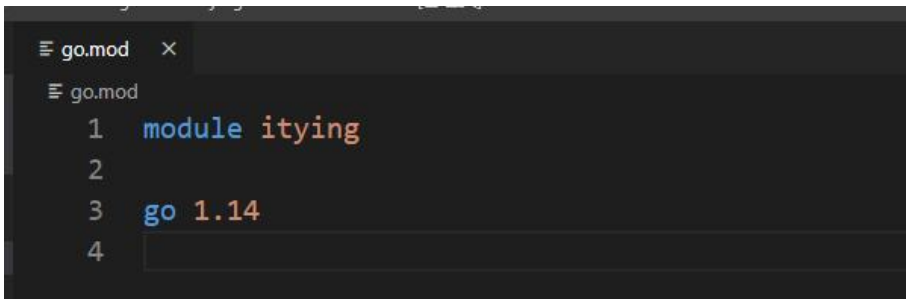
在 Golang1.11 版本之前如果我们要自定义包的话必须把项目放在 GOPATH 目录。Go1.11 版本之后无需手动配置环境变量，使用 go mod 管理项目，也不需要非得把项目放到 GOPATH 指定目录下，你可以在你磁盘的任何位置新建一个项目 ，Go1.13 以后可以彻底不要 GOPATH

了。

1、go mod init 初始化项目

实际项目开发中我们首先要在我们项目目录中用 go mod 命令生成一个 go.mod 文件管理我们项目的依赖。

比如我们的 golang 项目文件要放在了 itying 这个文件夹，这个时候我们需要在 itying 文件夹里面使用 go mod 命令生成一个 go.mod 文件



2、go mod 其他命令

```
D:\go_demo\demo18_pacakge\itying>go mod
Go mod provides access to operations on modules.

Note that support for modules is built into all the go commands,
not just 'go mod'. For example, day-to-day adding, removing, upgrading,
and downgrading of dependencies should be done using 'go get'.
See 'go help modules' for an overview of module functionality.

Usage:

        go mod <command> [arguments]

The commands are:

        download    download modules to local cache
        edit        edit go.mod from tools or scripts
        graph       print module requirement graph
        init        initialize new module in current directory
        tidy        add missing and remove unused modules
        vendor      make vendored copy of dependencies
        verify      verify dependencies have expected content
        why         explain why packages or modules are needed

Use "go help mod <command>" for more information about a command.
```

download download modules to local cache (下载依赖的 module 到本地 cache))

edit edit go.mod from tools or scripts (编辑 go.mod 文件)

graph print module requirement graph (打印模块依赖图))

init initialize new module in current directory (再当前文件夹下初始化一个新的 module，创建 go.mod 文件))

tidy add missing and remove unused modules (增加丢失的 module，去掉未用的 module)

vendor make vendored copy of dependencies (将依赖复制到 vendor 下)

verify verify dependencies have expected content (校验依赖 检查下载的第三方库有没有本地修改，如果有修改，则会返回非 0，否则验证成功。)

why explain why packages or modules are needed (解释为什么需要依赖)

三、Golang 中自定义包

包（package）是多个 Go 源码的集合，一个包可以简单理解为一个存放多个.go 文件的文件夹。该文件夹下面的所有 go 文件都要在代码的第一行添加如下代码，声明该文件归属的包。

```
package 包名
```

注意事项：

- 一个文件夹下面直接包含的文件只能归属一个 package，同样一个 package 的文件不能在多个文件夹下。
- 包名可以不和文件夹的名字一样，包名不能包含 - 符号。
- 包名为 main 的包为应用程序的入口包，这种包编译后会得到一个可执行文件，而编译不包含 main 包的源代码则不会得到可执行文件。

1、定义一个包

如果想在一个包中引用另外一个包里的标识符（如变量、常量、类型、函数等）时，该标识符必须是对外可见的（public）。在 Go 语言中只需要将标识符的首字母大写就可以让标识符对外可见了。

1、定义一个包名为 calc 的包，代码如下：

```
package calc

//首字母大小表示公有，首字母小写表示私有

var a = 100   //私有变量

var Age = 20 //公有变量

func Add(x, y int) int {

      return x + y

}

func Sum(x, y int) int {

      return x - y

}
```

2、main.go 中引入这个包

访问一个包里面的公有属性方法的时候需要通过**包名称.**去访问

```
package main
```

```
import (
	"fmt"
	"itying/calc"
)
func main() {
	c := calc.Add(10, 20)
	fmt.Println(c)
}
```

2、导入一个包

单行导入

单行导入的格式如下：

```
import "包 1"
import "包 2"
```

多行导入

多行导入的格式如下：

```
import (
    "包 1"
    "包 2"
)
```

匿名导入包

如果只希望导入包，而不使用包内部的数据时，可以使用匿名导入包。具体的格式如下：

```
import _ "包的路径"
```

匿名导入的包与其他方式导入的包一样都会被编译到可执行文件中。

自定义包名

在导入包名的时候，我们还可以为导入的包设置别名。通常用于导入的包名太长或者导入的包名冲突的情况。具体语法格式如下：

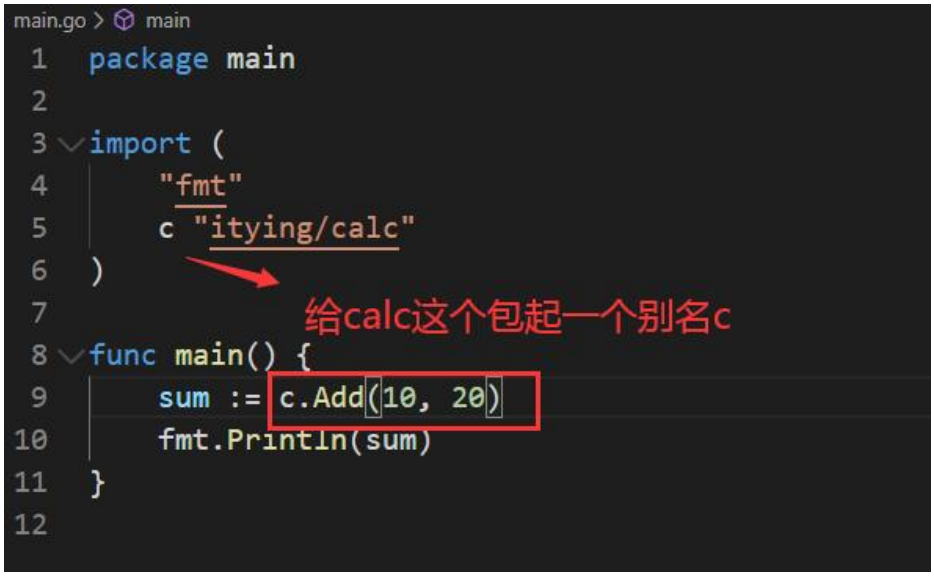
```
import 别名 "包的路径"
```

单行引入定义别名：

```
import c "itying/calc"
```

多行引入定义别名：

```
import (
    "fmt"
    c "itying/calc"
)
```



四、Golang 中 init()初始化函数

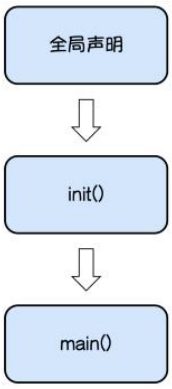
init()函数介绍

在 Go 语言程序执行时导入包语句会自动触发包内部 init()函数的调用。需要注意的是： init()函数没有参数也没有返回值。 init()函数在程序运行时自动被调用执行，不能在代码中主动调用它。

包初始化执行的顺序如下图所示：

包中init函数的执行时机

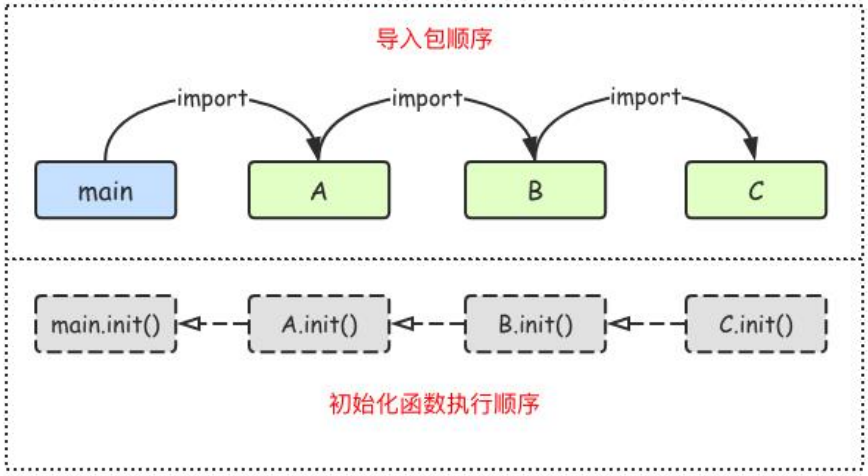
```go
package main

import "fmt"

var x int8 = 10

const pi = 3.14

func init() {
    fmt.Println(x)
}

func main() {
    fmt.Println("Hello")
}
```



init()函数执行顺序

Go 语言包会从 main 包开始检查其导入的所有包，每个包中又可能导入了其他的包。Go 编译器由此构建出一个树状的包引用关系，再根据引用顺序决定编译顺序，依次编译这些包的代码。

在运行时，被最后导入的包会最先初始化并调用其 init()函数， 如下图示：



五、Golang 中使用第三方包

我们可以在 https://pkg.go.dev/ 查找看常见的 golang 第三方包

1、初始化项目

```
go mod init 项目名
```

2、下载安装这个包（非必须）

比如前面给大家演示的解决 float 精度损失的包 decimal

https://github.com/shopspring/decimal

提示：此命令需要 cd 到项目里面执行

```
go get github.com/shopspring/decimal
```

3、看文档使用这个包

包安装完毕后我们就可以看文档使用这个包了，引入包以后可以使用 go mod tidy 增加丢失的 module 去掉未用的 module

4、go mod tidy 下载丢失的包

go mod tidy 增加丢失的 module 去掉未用的 module （**推荐**）

```
go mod tidy
```