李文周的博客

JPG程序员/全栈开发 -- 专注互联网技术，相信代码改变世界。Go语言学习QQ群：645090316 公众号：李文周

Go语言基础之包

2017年6月24日 | Golang |6094 阅读

在工程化的Go语言开发项目中，Go语言的源码复用是建立在包（package）基础之上的。本文介绍了Go语言中如何定义包、如何导出包的内容及如何导入其他包。

Go语言的包（package）

包介绍

`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`、`os`、`io`等。

定义包

我们还可以根据自己的需要创建自己的包。一个包可以简单理解为一个存放`.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。

可见性

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。

举个例子， 我们定义一个包名为 `pkg2` 的包，代码如下：

```
package pkg2

import "fmt"

// 包变量可见性

var a = 100 // 首字母小写，外部包不可见，只能在当前包内使用

// 首字母大写外部包可见，可在其他包中使用
const Mode = 1

type person struct { // 首字母小写，外部包不可见，只能在当前包内使用
    name string
}

// 首字母大写，外部包可见，可在其他包中使用
func Add(x, y int) int {
    return x + y
}

func age() { // 首字母小写，外部包不可见，只能在当前包内使用
    var Age = 18 // 函数局部变量，外部包不可见，只能在当前函数内使用
    fmt.Println(Age)
}
```

结构体中的字段名和接口中的方法名如果首字母都是大写，外部包可以访问这些字段和方法。例如：

```
type Student struct {
    Name  string //可在包外访问的方法
    class string //仅限包内访问的字段
}

type Payer interface {
    init() //仅限包内访问的方法
    Pay()  //可在包外访问的方法
}
```

包的导入

要在代码中引用其他包的内容，需要使用 `import` 关键字导入使用的包。具体语法如下：

```
import "包的路径"
```

注意事项：

- import导入语句通常放在文件开头包声明语句的下面。
- 导入的包名需要使用双引号包裹起来。
- 包名是从 $GOPATH/src/后开始计算的，使用 /进行路径分隔。
- Go语言中禁止循环导入包。

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"
)
```

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"
```

单行导入方式定义别名：

```
import "fmt"
import m "github.com/Q1mi/studygo/pkg_test"

func main() {
    fmt.Println(m.Add(100, 200))
    fmt.Println(m.Mode)
}
```

多行导入方式定义别名：

```
import (
    "fmt"
    m "github.com/Q1mi/studygo/pkg_test"
 )

func main() {
    fmt.Println(m.Add(100, 200))
    fmt.Println(m.Mode)
}
```

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。

init()初始化函数

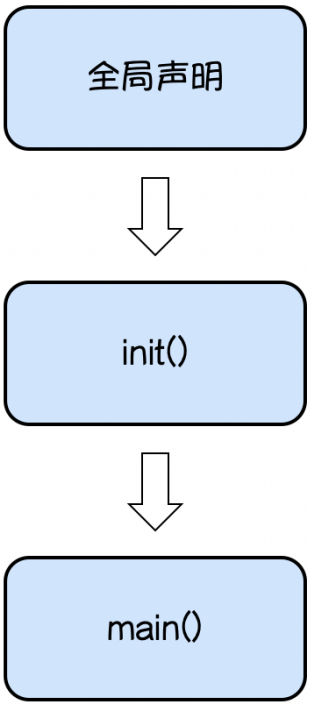
init()函数介绍

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

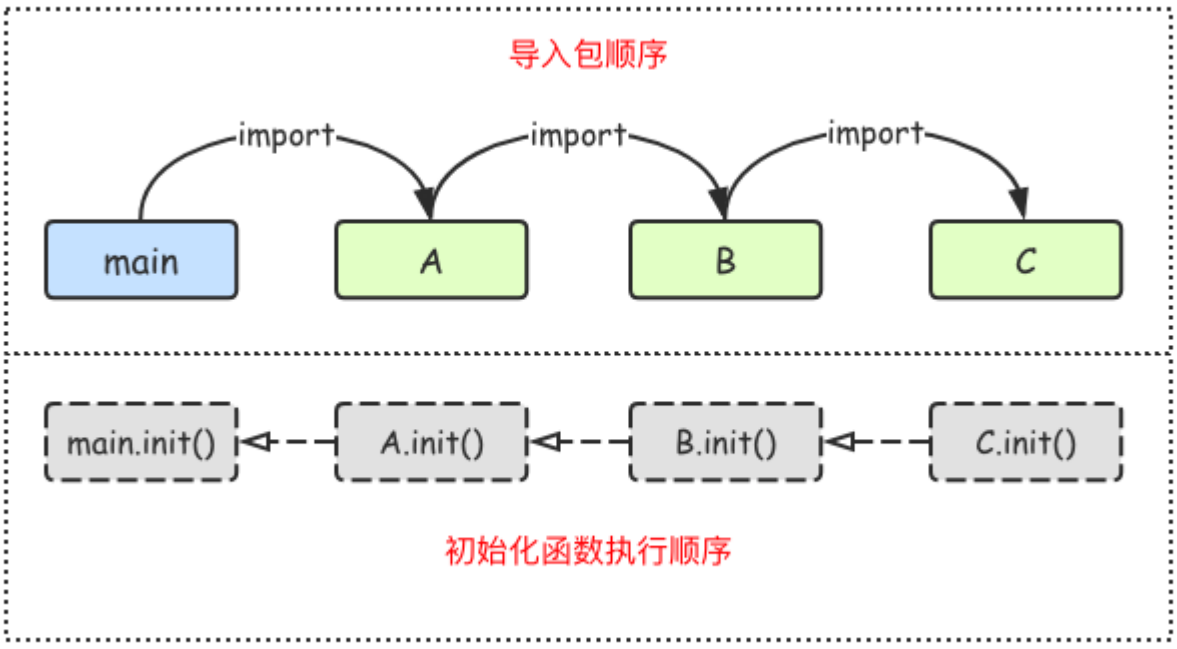
```go
package main

import "fmt"

var x int8 = 10

const pi = 3.14

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

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



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：



练习题

1. 编写一个`calc`包实现加减乘除四个功能函数，在`snow`这个包中导入并使用加减乘除四个函数实现数学运算。