> 慕课网慕课教程 7. go 语言编码规范涵盖海量编程基础技术教程，以图文图表的形式，把晦涩难懂的编程专业用语，以通俗易懂的方式呈现给用户。

1. 代码规范不是强制的，也就是你不遵循代码规范写出来的代码运行也是完全没有问题的
2. 代码规范目的是方便团队形成一个统一的代码风格，提高代码的可读性，规范性和统一性。本规范将从命名规范，注释规范，代码风格和 Go 语言提供的常用的工具这几个方面做一个说明。
3. 规范并不是唯一的，也就是说理论上每个公司都可以制定自己的规范，不过一般来说整体上规范差异不会很大。

命名是代码规范中很重要的一部分，统一的命名规则有利于提高的代码的可读性，好的命名仅仅通过命名就可以获取到足够多的信息。
a. 当命名（包括常量、变量、类型、函数名、结构字段等等）以一个大写字母开头，如：Group1，那么使用这种形式的标识符的对象就**可以被外部包的代码所使用**（客户端程序需要先导入这个包），这被称为导出（像面向对象语言中的 public）；
b. 命名如果以小写字母开头，则对包外是不可见的，但是他们在整个包的内部是可见并且可用的（像面向对象语言中的 private ）

1.1 包名：package

保持 package 的名字和目录保持一致，尽量采取有意义的包名，简短，有意义，尽量和标准库不要冲突。包名应该为**小写**单词，不要使用下划线或者混合大小写。

```
package model
package main
```

1.2 文件名

尽量采取有意义的文件名，简短，有意义，应该为**小写**单词，使用**下划线**分隔各个单词。

1.3 结构体命名

- 采用驼峰命名法，首字母根据访问控制大写或者小写
- struct 申明和初始化格式采用多行，例如下面：

```
type User struct{
    Username  string
    Email     string
}


u := User{
    Username: "bobby",
    Email:    "bobby@imooc.com",
}
```

1.4 接口命名

- 命名规则基本和上面的结构体类型
- 单个函数的结构名以 “er” 作为后缀，例如 Reader , Writer 。

```
type Reader interface {
        Read(p []byte) (n int, err error)
}
```

1.5 变量命名

- 和结构体类似，变量名称一般遵循驼峰法，首字母根据访问控制原则大写或者小写，但遇到特有名词时，需要遵循以下规则：
- 如果变量为私有，且特有名词为首个单词，则使用小写，如 apiClient
- 其它情况都应当使用该名词原有的写法，如 APIClient、repoID、UserID
- 错误示例：UrlArray，应该写成 urlArray 或者 URLArray
- 若变量类型为 bool 类型，则名称应以 Has, Is, Can 或 Allow 开头

```
var isExist bool
var hasConflict bool
var canManage bool
var allowGitHook bool
```

1.6 常量命名

常量均需使用全部大写字母组成，并使用下划线分词

如果是枚举类型的常量，需要先创建相应类型：

```
type Scheme string

const (
    HTTP  Scheme = "http"
    HTTPS Scheme = "https"
)
```

Go 提供 C 风格的`/* */`块注释和 C ++ 风格的`//`行注释。行注释是常态；块注释主要显示为包注释，但在表达式中很有用或禁用大量代码。

- 单行注释是最常见的注释形式，你可以在任何地方使用以 // 开头的单行注释
- 多行注释也叫块注释，均已以 /* 开头，并以 */ 结尾，且不可以嵌套使用，多行注释一般用于包的文档描述或注释成块的代码片段

go 语言自带的 godoc 工具可以根据注释生成文档，生成可以自动生成对应的网站（golang.org 就是使用 godoc 工具直接生成的），注释的质量决定了生成的文档的质量。每个包都应该有一个包注释，在 package 子句之前有一个块注释。对于多文件包，包注释只需要存在于一个文件中，任何一个都可以。包评论应该介绍包，并提供与整个包相关的信息。它将首先出现在`godoc`页面上，并应设置下面的详细文档。

2.1 包注释

每个包都应该有一个包注释，一个位于 package 子句之前的块注释或行注释。包如果有多个 go 文件，只需要出现在一个 go 文件中（一般是和包同名的文件）即可。 包注释应该包含下面基本信息 (请严格按照这个顺序，简介，创建人，创建时间）：

- 包的基本简介（包名，简介）
- 创建者，格式： 创建人： rtx 名
- 创建时间，格式：创建时间： yyyyMMdd

2.2 结构（接口）注释

每个自定义的结构体或者接口都应该有注释说明，该注释对结构进行简要介绍，放在结构体定义的前一行，格式为： 结构体名， 结构体说明。同时结构体内的每个成员变量都要有说明，该说明放在成员变量的后面（注意对齐），实例如下：

```
type User struct{
    Username  string
    Email     string
}
```

2.3 函数（方法）注释

每个函数，或者方法（结构体或者接口下的函数称为方法）都应该有注释说明，函数的注释应该包括三个方面（严格按照此顺序撰写）：

- 简要说明，格式说明：以函数名开头，“，”分隔说明部分
- 参数列表：每行一个参数，参数名开头，“，”分隔说明部分
- 返回值： 每行一个返回值

```
func NewAttrModel(ctx *common.Context) *AttrModel {
}
```

2.4 代码逻辑注释

对于一些关键位置的代码逻辑，或者局部较为复杂的逻辑，需要有相应的逻辑说明，方便其他开发者阅读该段代码，实例如下：

```
xxxxx
xxxxxxx
xxxxxxx
```

2.5 注释风格

统一使用中文注释，对于中英文字符之间严格使用空格分隔， 这个不仅仅是中文和英文之间，英文和中文标点之间也都要使用空格分隔，例如：

上面 Redis 、 id 、 DB 和其他中文字符之间都是用了空格分隔。

- 建议全部使用单行注释
- 和代码的规范一样，单行注释不要过长，禁止超过 120 字符。

import 在多行的情况下，goimports 会自动帮你格式化，但是我们这里还是规范一下 import 的一些规范，如果你在一个文件里面引入了一个 package，还是建议采用如下格式：

如果你的包引入了三种类型的包，标准库包，程序内部包，第三方包，建议采用如下方式进行组织你的包：

```
import (
    "encoding/json"
    "strings"

    "myproject/models"
    "myproject/controller"
    "myproject/utils"

    "github.com/astaxie/beego"
    "github.com/go-sql-driver/mysql"
)
```

有顺序的引入包，不同的类型采用空格分离，第一种实标准库，第二是项目包，第三是第三方包。
在项目中不要使用相对路径引入包：

```
import “../net”


import “github.com/repo/proj/src/net”
```

但是如果是引入本项目中的其他包，最好使用相对路径。

- 错误处理的原则就是不能丢弃任何有返回 err 的调用，不要使用 _ 丢弃，必须全部处理。接收到错误，要么返回 err，或者使用 log 记录下来
- 尽早 return：一旦有错误发生，马上返回
- 尽量不要使用 panic，除非你知道你在做什么
- 错误描述如果是英文必须为小写，不需要标点结尾

- 采用独立的错误流进行处理

```
if err != nil {

} else {

}


if err != nil {

    return
}
```