

break	default	func	interface	select
case	defer	Go	map	Struct
chan	else	Goto	package	Switch
const	fallthrough	if	range	Type
continue	for	import	return	Var

[代码规范]Go语言编码规范指导



千言千语
跟着小新学IT，简单易懂很nice

关注她

51 人赞同了该文章

本规范旨在为日常Go项目开发提供一个代码的规范指导，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。该规范参考了 go 语言官方代码的风格制定。

一、命名规范

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。

Go在命名时以字母a到Z或a到z或下划线开头，后面跟着零或更多的字母、下划线和数字(0到9)。Go不允许在命名时使用@、\$和%等标点符号。Go是一种区分大小写的编程语言。因此，Manpower和manpower是两个不同的命名。

- 1. 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）；
- 2. 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、包命名：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 demo

package main
```

2、文件命名

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。

```
my_test.go
```

3、结构体命名



- struct 申明和初始化格式采用多行，例如下面：

```
// 多行申明
type User struct{
    Username string
    Email    string
}

// 多行初始化
u := User{
    Username: "astaxie",
    Email:    "astaxie@gmail.com",
}
```

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)
}
```

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
```

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

```
const APP_VER = "1.0"
```

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"
)
```

7、关键字

下面的列表显示了Go中的保留字。这些保留字不能用作常量或变量或任何其他标识符名称。



二、注释

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。

详细的如何写注释可以 参考：golang.org/doc/effective-go

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

例如 `util` 包的注释示例如下

```
// util 包， 该包包含了项目共用的一些常量，封装了项目中一些共用函数。
// 创建人： hanru
// 创建时间： 20190419
```

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：

```
// User ， 用户对象，定义了用户的基础信息
type User struct{
    Username string // 用户名
    Email    string // 邮箱
}
```

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）：



- 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

示例如下：

```
// NewAttrModel , 属性数据层操作类的工厂方法
// 参数:
//      ctx : 上下文信息
// 返回值:
//      属性操作类指针
func NewAttrModel(ctx *common.Context) *AttrModel {
}
```

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：

```
// 从 Redis 中批量读取属性，对于没有读取到的 id ， 记录到一个数组里面，准备从 DB 中读取
xxxxx
xxxxxxx
xxxxxxx
```

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：

```
// 从 Redis 中批量读取属性，对于没有读取到的 id ， 记录到一个数组里面，准备从 DB 中读取
```

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。

三、代码风格

1、缩进和折行

- 缩进直接使用 gofmt 工具格式化即可（gofmt 是使用 tab 缩进的）；
- 折行方面，一行最长不超过120个字符，超过的请使用换行展示，尽量保持格式优雅。

我们使用Goland开发工具，可以直接使用快捷键：ctrl+alt+L，即可。

2、语句的结尾

Go语言中是不需要类似于Java需要冒号结尾，默认一行就是一条数据

如果你打算将多个语句写在同一行，它们则必须使用；

3、括号和空格

括号和空格方面，也可以直接使用 gofmt 工具格式化（go 会强制左大括号不换行，换行会报语法错误），所有的运算符和操作数之间要留空格。

```
// 正确的方式
if a > 0 {
```



```
// 错误的方式
if a>0 // a, 0 和 > 之间应该空格
{      // 左大括号不可以换行，会报语法错误

}
```

4、import 规范

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：

```
import (
    "fmt"
)
```

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/utills"

    "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：

```
// 这是不好的导入
import "../net"

// 这是正确的做法
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。

5、错误处理

- 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

```
// 错误写法
if err != nil {
    // error handling
} else {
    // normal code
}
```

```
// error handling
return // or continue, etc.
}
// normal code
```

6、测试

单元测试文件名命名规范为 example_test.go 测试用例的函数名称必须以 Test 开头，例如：TestExample 每个重要的函数都要首先编写测试用例，测试用例和正规代码一起提交方便进行回归测试

四、常用工具

上面提到了很过规范， go 语言本身在代码规范性这方面也做了很多努力，很多限制都是强制语法要求，例如左大括号不换行，引用的包或者定义的变量不使用会报错，此外 go 还是提供了很多好用的工具帮助我们进行代码的规范，

gofmt 大部分的格式问题可以通过gofmt解决， gofmt 自动格式化代码，保证所有的 go 代码与官方推荐的格式保持一致，于是所有格式有关问题，都以 gofmt 的结果为准。

goimport 我们强烈建议使用 goimport ， 该工具在 gofmt 的基础上增加了自动删除和引入包。

```
go get golang.org/x/tools/cmd/goimports
```

go vet vet工具可以帮我们静态分析我们的源码存在的各种问题，例如多余的代码，提前return的逻辑，struct的tag是否符合标准等。

```
go get golang.org/x/tools/cmd/vet
```

使用如下：

```
go vet .
```

发布于 2019-04-22 10:36

[Go 语言](#) [代码规范](#) [Go 编程](#)

2 条评论

切换为时间排序

写下你的评论...

邓嘉文Jarvan2021-06-29

好家伙，该早点看

赞

移动的红烧肉👍2020-09-02

赞，入门必看

赞



Go 语言开发设计指北

Meng小... 发表于Debug...



Go 语言开发设计指北

Alftr... 发表于Go语言大...

关于GO语言明白

摘要：本文从Go的生态系统，编码风格，语言工具和使用案例等几方面进行了学习和探讨。Go后，很多公司特别是用Go语言重构产品的

华为云开发... 发