

# 编程项目训练营

<基础课程第二周>



GO







# 目录

第一节 Go Style Guide

第二节 Go mod

第三节 Go 命令行解析

作业



HELLO, WORLD

# Go Style Guide

- 任何语言都需要强调编码风格的一致性。只要是团队开发，每个人都以相同的方式编写代码就是至关重要的。这样大家才能方便地互相看懂和维护对方的代码。



呦，写BUG呢

# Go Style Guide

- 参考资料:

➡ [Go Code Review Comments](#)

➡ <https://github.com/golang/go/wiki/CodeReviewComments> [英文版]

➡ <https://www.zybuluo.com/wddpct/note/1264988> [中文版]

➡ Uber Go Style Guide [https://github.com/xxjwxc/uber\\_go\\_guide\\_cn](https://github.com/xxjwxc/uber_go_guide_cn) [中文版]

- 精读:

➡ [https://golang.org/doc/effective\\_go.html](https://golang.org/doc/effective_go.html) [英文版]

➡ <https://github.com/uber-go/guide/blob/master/style.md> [英文版]





# 目录

第一节 Go Style Guide

第二节 Go mod

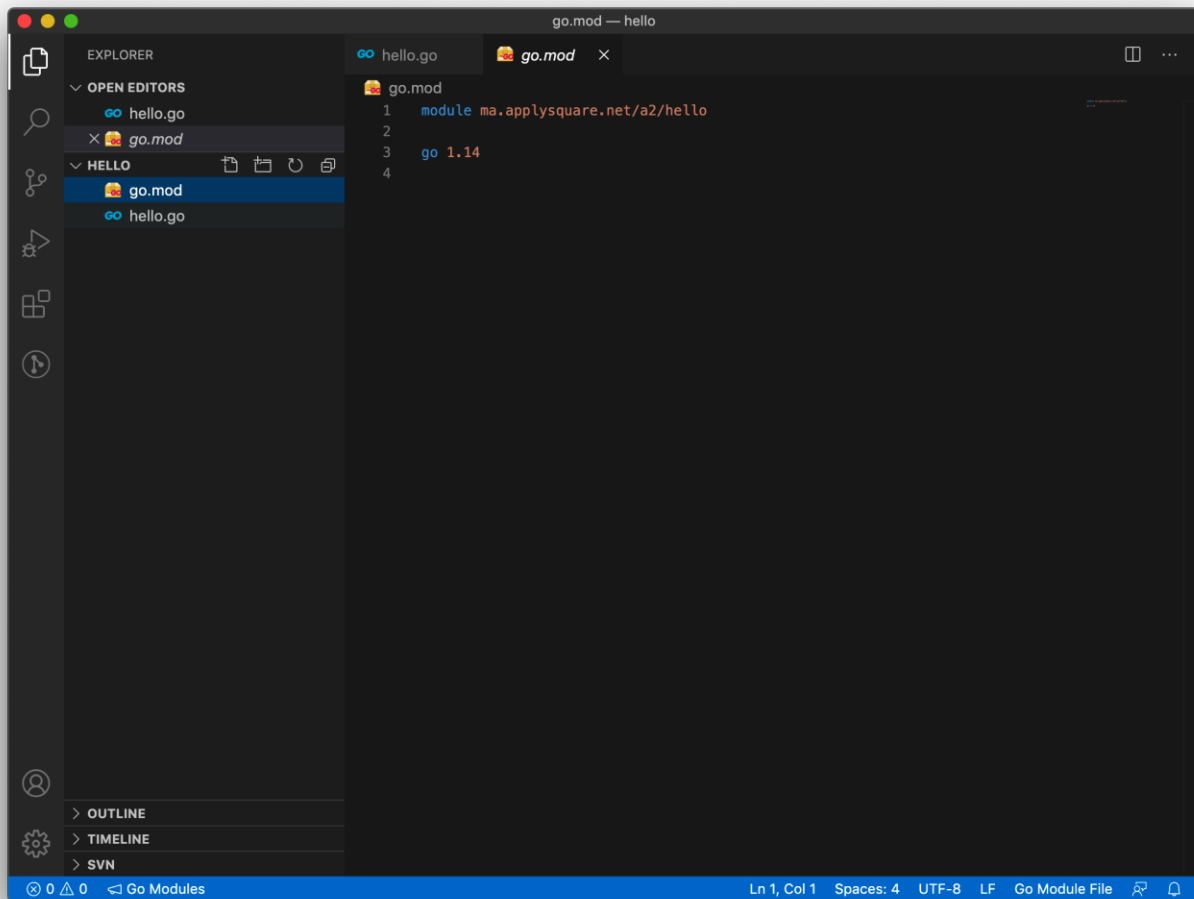
第三节 Go 命令行解析

作业



HELLO, WORLD

# Go mod



- Go.mod是Golang1.11版本新引入的官方包管理工具用于解决之前没有地方记录依赖包具体版本的问题，方便依赖包的管理。
- Go.mod其实就是一个Modules，关于Modules的官方定义为：
- Modules是相关Go包的集合，是源代码交换和版本控制的单元。go命令直接支持使用Modules，包括记录和解析对其他模块的依赖性。Modules替换旧的基于GOPATH的方法，来指定使用哪些源文件。

# Go mod



## macOS 或 Linux

打开你的终端并执行

```
$ export G0111MODULE=on  
$ export GOPROXY=https://goproxy.cn
```

或者

```
$ echo "export G0111MODULE=on" >> ~/.profile  
$ echo "export GOPROXY=https://goproxy.cn" >> ~/.profile  
$ source ~/.profile
```

完成。



# Go mod

## go mod 使用方法

- 初始化模块

```
go mod init <项目模块名称>
```

- 依赖关系处理,根据go.mod文件

```
go mod tidy
```

- 将依赖包复制到项目下的 vendor目录。

```
go mod vendor
```

如果包被屏蔽(墙),可以使用这个命令,随后使用go build -mod=vendor编译

- 显示依赖关系

```
go list -m all
```

- 显示详细依赖关系

```
go list -m -json all
```

- 下载依赖

```
go mod download [path@version]
```

[path@version]是非必写的

- cd ~/go/src/ma.applysquare.net/a2/hello
- go mod init

```
→ hello cd ~/go/src/ma.applysquare.net/a2/hello
→ hello ls
→ hello go mod init
go: creating new go.mod: module ma.applysquare.net/a2/hello
→ hello ls
go.mod
→ hello go mod tidy
→ hello cat go.mod
module ma.applysquare.net/a2/hello

go 1.14
```





# 目录

第一节 Go Style Guide

第二节 Go mod

第三节 Go 命令行解析

作业



HELLO, WORLD

# 命令行解析

```
→ ~ go
Go is a tool for managing Go source code.

Usage:

    go <command> [arguments]

The commands are:

    bug          start a bug report
    build         compile packages and dependencies
    clean        remove object files and cached files
    doc          show documentation for package or symbol
    env          print Go environment information
    fix          update packages to use new APIs
    fmt          gofmt (reformat) package sources
    generate      generate Go files by processing source
    get          add dependencies to current module and install them
    install      compile and install packages and dependencies
    list         list packages or modules
    mod          module maintenance
    run          compile and run Go program
    test         test packages
    tool         run specified go tool
    version      print Go version
    vet          report likely mistakes in packages

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

Additional help topics:

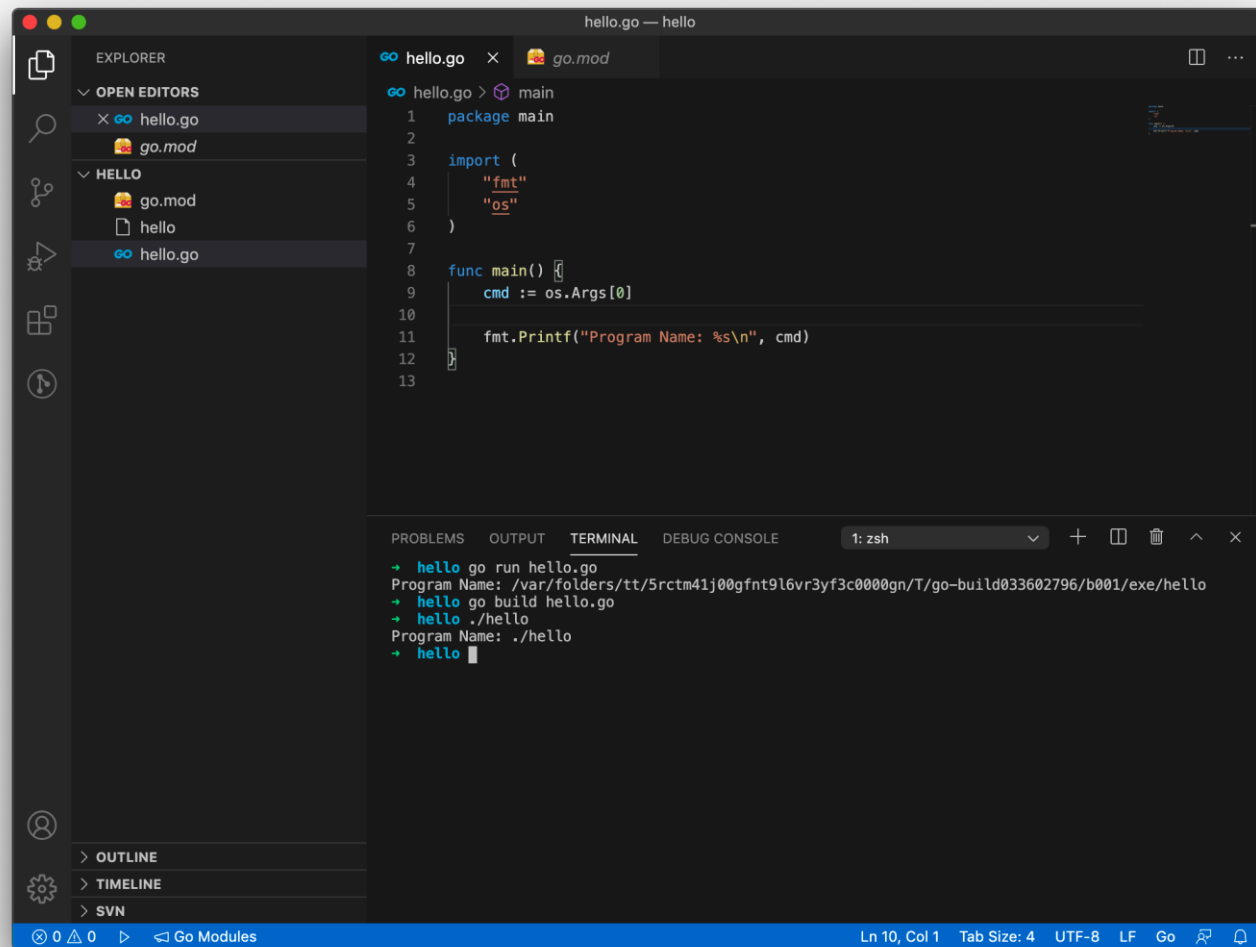
    buildmode    build modes
    c            calling between Go and C
    cache        build and test caching
    environment  environment variables
    filetype     file types
    go.mod       the go.mod file
    gopath       GOPATH environment variable
    gopath-get   legacy GOPATH go get
    goproxy      module proxy protocol
    importpath   import path syntax
    modules      modules, module versions, and more
    module-get   module-aware go get
    module-auth  module authentication using go.sum
    module-private module configuration for non-public modules
    packages     package lists and patterns
    testflag     testing flags
    testfunc     testing functions

Use "go help <topic>" for more information about that topic.
```

- 使用命令程序对程序员来说很常见，就算是前端工程师或者开发gui的，也需要使用命令行来编译程序或者打包程序
- 熟练使用命令行工具能极大的提高开发效率，linux自带的命令行工具都非常的有用。
- 学习了Go语言，我们也可以编写自己的命令程序。
- 左侧截图是以go命令为例，展示了其参数



# 命令行解析



The screenshot shows the VS Code editor interface. The Explorer sidebar on the left shows the project structure with files `hello.go` and `go.mod`. The main editor window displays the `hello.go` file with the following code:

```
1 package main
2
3 import (
4     "fmt"
5     "os"
6 )
7
8 func main() {
9     cmd := os.Args[0]
10
11     fmt.Printf("Program Name: %s\n", cmd)
12 }
13
```

The bottom panel shows the TERMINAL output for the command `go run hello.go`:

```
→ hello go run hello.go
Program Name: /var/folders/tt/5rctm41j00gfnt9l6vr3yf3c0000gn/T/go-build033602796/b001/exe/hello
→ hello go build hello.go
→ hello ./hello
Program Name: ./hello
→ hello
```

- 获取命令行名称和参数的示例

- 参考资料:

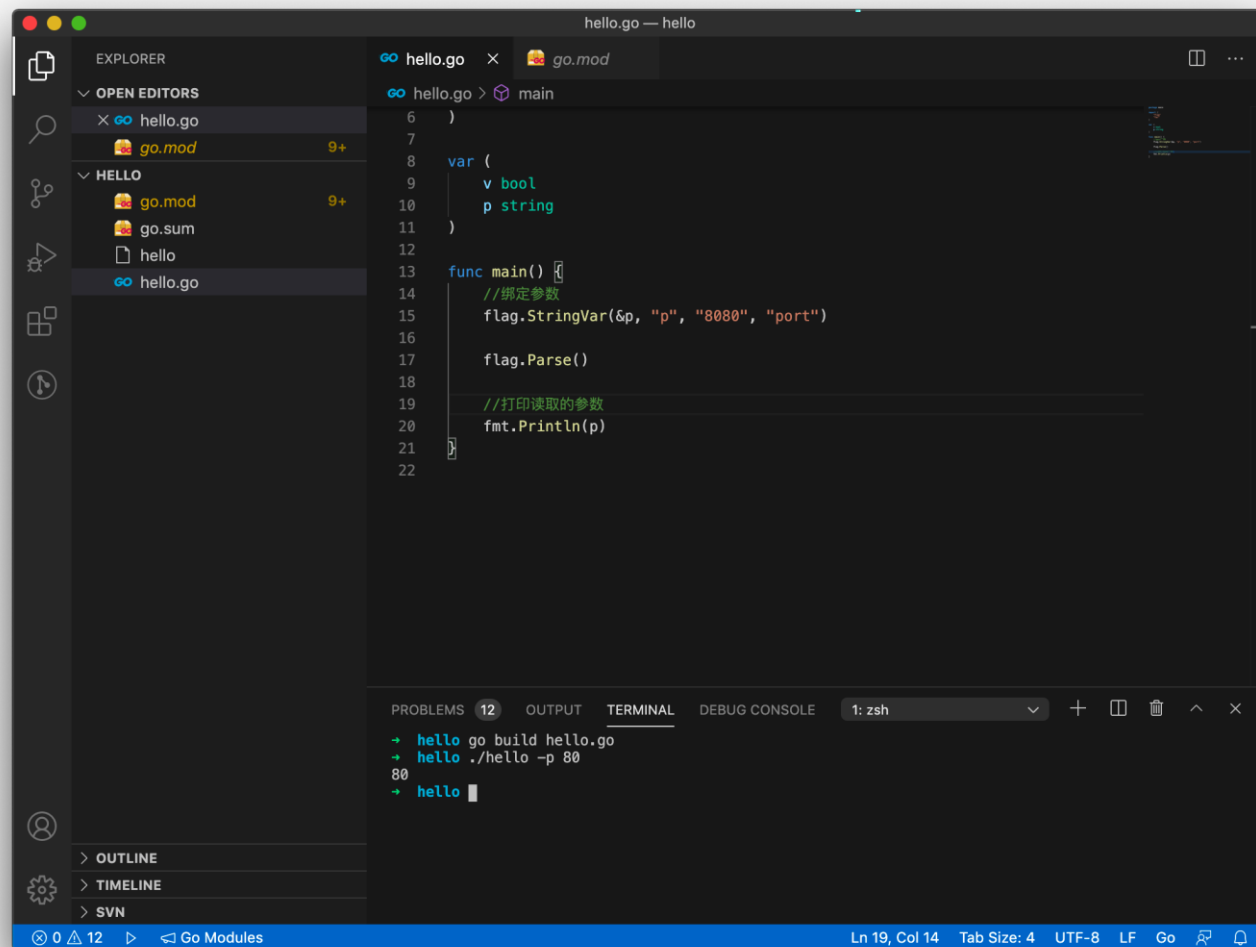
<https://studygolang.com/articles/7857>

# 命令行解析

- Flag 包实现了命令行标志解析。

- 参考资料：

<https://cloud.tencent.com/developer/section/1141707>







# 目录

第一节 Go Style Guide

第二节 Go mod

第三节 Go 命令行解析

作业



HELLO, WORLD

新建一个Go项目，使用go mod进行包。完成一个命令程序，接受h,type参数。输入h时候，命令行显示使用帮助；type值为1时输出一句唐诗，为2时输出一句宋词。

提交形式 >>>

➡ 教学立方平台第二周作业

➡ PDF文档

- Fork作业仓库
- Github代码截图，并分享自己作业的Github代码地址。
- 虚拟机上执行main.go的截图