练识课堂 -- Go语言资深工程师培训课程

第一章 Go语言介绍和环境安装

1 Go语言介绍

> **Go 语言官方介绍**
>
> Go 语言是有谷歌推出的一门编程语言。
>
> Go 是一个开源的编程语言，它能让构造简单、可靠且高效的软件变得容易。

> **Go 语言主要开发者**
>
> Go 是从2007年末由Robert Griesemer, Rob Pike, Ken Thompson主持开发，后来还加入了Ian Lance Taylor, Russ Cox等人，并最终于2009年11月开源。

- Rob Pike (UTF-8, Plan 9 from Bell Labs)
- Ken Thompson (UNIX, B, C++, Plan 9)
- Robert Griesemer (HotSpot, V8)

> **Go语言的几个里程碑**
>
> - 2007.09 Go语言设计草稿在白板上诞生
> - 2008.01 Ken开发了Go语言编译器把Go代码编译成C代码
> - 2009 Go语言诞生
> - 2016.01 Go 1.5 版本中实现了自举，不再依赖C编译器
> - 2017.02 Go 1.8 版本发布，大幅度提升GC效率

> **Go 语言特点**
>
> - 简洁、快速、安全
> - 并行、有趣、开源
> - 内存管理、数组安全、编译迅速

> **Go 语言方向**
>
> - 网络编程领域
> - 区块链开发领域
> - 高性能分布式系统领域

2 环境安装

> **Go 语言的环境安装:**
>
> - Go安装包下载网址(Go语言中文网)：https://studygolang.com/dl
> - 根据操作系统选择对应的安装包（Mac、Linux、Windows）。
> - 不要在安装路径中出现中文。

> **Go 语言开发工具:**
>
> - GoLand
>   - GoLand 是 Jetbrains 家族的 Go 语言 IDE，有 30 天的免费试用期。
>   - 下载安装网址：https://www.jetbrains.com/go/
>   - 根据操作系统选择对应的安装包（Mac、Linux、Windows）下载对应的软件。
> - LiteIDE
>   - LiteIDE 是一款开源、跨平台的轻量级 Go 语言集成开发环境（IDE）。
>   - 下载安装网址：https://sourceforge.net/projects/liteide/files/
>   - 根据操作系统选择对应的安装包（Linux、Windows）下载对应的软件。
> - 其他开发工具 (在IDE安装插件)
>   - Eclipse
>   - VS Code

第二章 Go语言程序

1 工作区

> - 工作区是Go中的一个对应于特定工程的目录，其包括src，pkg，bin三个目录
>   - src：用于以代码包的形式组织并保存Go源码文件。（比如：.go .c .h .s等）
>   - pkg：用于存放经由go install命令构建安装后的代码包（包含Go库源码文件）的“.a”归档文件。
>   - bin：与pkg目录类似，在通过go install命令完成安装后，保存由Go命令源码文件生成的可执行文件。
>   - 目录src用于包含所有的源代码，是Go命令行工具一个强制的规则，而pkg和bin则无需手动创建，如果必要Go命令行工具在构建过程中会自动创建这些目录。
>   - 只有当环境变量 GOPATH 中只包含一个工作区的目录路径时，go install命令才会把命令源码安装到当前工作区的bin目录下。若环境变量 GOPATH 中包含多个工作区的目录路径，像这样执行go install命令就会失效，此时必须设置环境变量GOBIN。
>
> ```
> go install: no install location for .go files listed on command line
> (GOBIN not set)
> ```

- 工作区如何设置(linux mac版)
    - 工作区设置路径为环境变量中的GOPATH

```
#mac设置GOPATH
vim .bash_profile
export GOPATH=/Users/lianshi/GoWork/Public:/Users/lianshi/GoWork/Company
```

```
#linux设置GOPATH
vim .bashrc
export GOROOT=$HOME/go
export PATH=$PATH:$GOROOT/bin
export GOPATH=$HOME/workspace/go
export PATH=$PATH:$GOPATH/bin
```

- 工作区如何设置(windows版)
    - 工作区设置路径为环境变量中的GOPATH

```
我的电脑 --》 右击属性 --》 高级系统设置 --》 环境变量 --》 系统环境变量 --》 添加GOPATH
```

2 Hello world

下面就用IDE工具，开发第一个GO程序。

Go 语言源文件的扩展是 .go

具体步骤如下：s

```
package main

import "fmt"

func main() {
fmt.Println("Hello, World!")
fmt.Println("性感法师在线教学~")
}
```

3 编译过程

要执行 Go 语言代码可以使用命令或IDE来完成编译。

命令如下：

- 编译命令：go build hello.go
- 编译并运行命令：go run hello.go

4 常用命令行

go help

- 获取对应命令的帮助文档，可以获取到对应命令的作用以及对应参数

```
go help build
```

go version

- 获取系统安装go语言版本号

go build

- 编译项目，使其打包成可运行程序，配合参数可以进行交叉编译
- 标准格式
  - go build [-o output] [-i] [build flags] [packages]
  - -o 参数决定了编译后文件名称，例如我们要程序main.go编译后程序名为hello，我们可以执行以下命令

```
go build -o hello main.go
```

- -i install 安装作为目标的依赖关系的包(用于增量编译提速)，一般很少使用。
- 编译参数一般并不会添加，以下列举几个，详细信息可以使用go help build 获取

附加参数	备注
-v	编译时显示包名
-p n	开启并发编译，默认情况下该值为 CPU 逻辑核数
-a	强制重新构建
-n	打印编译时会用到的所有命令，但不真正执行
-x	打印编译时会用到的所有命令

- packages
  - 所编译的包名，如果不填写默认为编译当前路径下的入口文件，文件名称默认为当前文件夹名称

交叉编译

- go语言向下支持C语言，可以在go语言中直接编写C语言代码
- 但是在编译时，必须支持C语言
- Mac上编译Linux可执行二进制文件

```
CGO_ENABLED=0 GOOS=linux GOARCH=amd64 go build main.go
```

- Mac上编译Windows可执行二进制文件

```
CGO_ENABLED=0 GOOS=windows GOARCH=amd64 go build main.go
```

- Linux上编译Mac可执行二进制文件

```
CGO_ENABLED=0 GOOS=darwin GOARCH=amd64 go build main.go
```

- Linux上编译Windows可执行二进制文件

```
CGO_ENABLED=0 GOOS=windows GOARCH=amd64 go build main.go
```

- Windows上编译Mac可执行二进制文件

```
SET CGO_ENABLED=0 SET GOOS=darwin SET GOARCH=amd64 go build main.go
```

- Windows上编译Linux可执行二进制文件

```
SET CGO_ENABLED=0 SET GOOS=linux SET GOARCH=amd64 go build main.go
```

- 交叉编译参数含义
  - CGO_ENABLED 是否使用cgo编译，0为不使用，1为使用，使用cgo进行交叉编译时需要编译机器安装对应的cgo程序
  - GOOS 目标操作系统标识，windows对应Windows操作系统exe可执行文件，darwin对应Mac可执行文件，linux对应Linux可执行文件，freebsd对应UNIX系统
  - GOARCH 目标可执行程序操作系统构架，包括 386，amd64，arm
  - go build 后接所编译程序的入口文件

go install

- 编译并安装项目
- 标准格式
  - go install [-i] [build flags] [packages]
  - 参数与用法与go build类似

go doc

- 获取go函数帮助文档
- 命令行形式获取某个包的介绍以及包下所有可用的公共方法列表

```
go doc strconv
```

- 命令行形式获取某个方法的文档

```
go doc strconv.Itoa
```

- 使用网页形式查看帮助文档

```
godoc -http=localhost:6060
```

- 在浏览器中输入：http://localhost:6060/ 可以查看对应的文档信息

go env

- 查看当前系统内go相关的环境变量信息

go test

- Go语言自带的测试工具，会自动读取源码目录下面名为 *_test.go 的文件，生成并运行测试用的可执行文件
- 原则
  - 文件名必须是 *_test.go* 结尾的，这样在执行 go test 的时候才会执行到相应的代码
  - 必须 import testing 这个包
- 可执行测试
  - 原则
    - 所有的测试用例函数必须是 Test 开头
    - 测试用例会按照源代码中写的顺序依次执行
    - 测试函数 TestXxx() 的参数是 testing.T ，我们可以使用该类型来记录错误或者是测试状态
    - 测试格式: func TestXxx (t *testing.T) , Xxx 部分可以为任意的字母数字的组合，但是首字母不能是小写字母[a-z]
    - 函数中通过调用 testing.T 的 Error, Errorf, FailNow, Fatal, FatalIf 方法，说明测试不通过，调用 Log 方法用来记录测试的信息。
  - 创建测试文件class_test.go

```
package main

import (
  "testing"
  "time"
)

func TestHelloWorld(t *testing.T)  {
  timestamp := time.Now().Unix()
  t.Log(timestamp)
}
```

- 执行命令行查看测试结果

```
go test -v class_test.go
```

- 结果

```
=== RUN   TestHelloWorld
--- PASS: TestHelloWorld (0.00s)
    class_test.go:10: 1571237717
PASS
ok   command-line-arguments  0.005s
```

+ === RUN   TestHelloWorld 表示开始运行名叫 TestHelloWorld 的测试用例
+ --- PASS: TestHelloWorld (0.00s) 表示已经运行完 TestHelloWorld 的测试用例，PASS 表示当前方法测试成功，如果是FAIL 表示当前方法测试失败，时间表示这个测试用例所使用的时间
+ ok   command-line-arguments 0.005s 表示整体测试结果，ok 表示所有被测试方法测试通过，如果是FAIL则表示测试失败，command-line-arguments 是测试用例需要用到的一个包名，0.005s 表示测试花费的时间。