

Backend Golang

Урок 2

Инструментарий (build, test, fmt), управление зависимостями (go mod)

Формат занятий

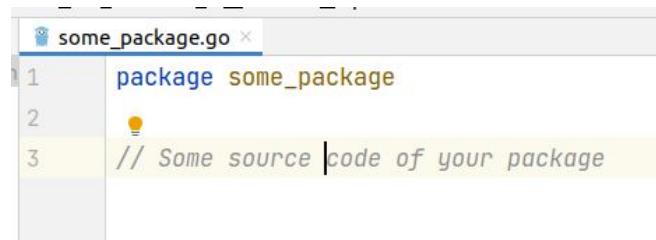
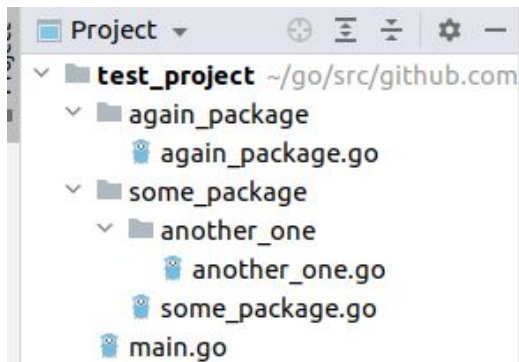
Все ключаем камеры (сложно коммуницировать с невидимками).

Занятие состоит 3-х уроков по 40 минут:

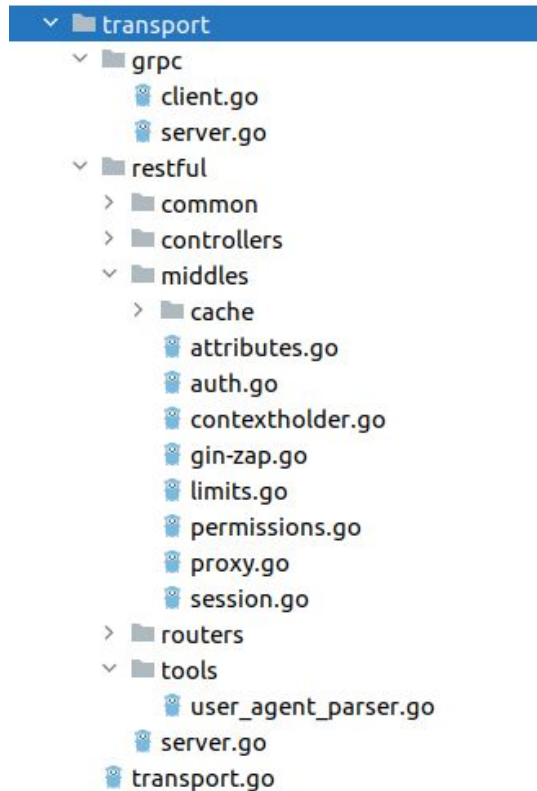
1. Разбор ДЗ, теория по новой теме;
2. Теория, практика;
3. Практика, закрепление материала.

Перерыв между уроками — 15 минут.

Строительный блок — package



Боевой пример



Программы на Go делятся на 2 типа:

1. Исполняемые
2. Библиотеки

Исполняемая программа содержит `package main` и `func main()`

```
package main

import (
    "log"

    "github.com/vrischmann/envconfig"
)

func main() {
    var conf struct {
        Name string `envconfig:"myName"`
    }
    if err := envconfig.Init(&conf); err != nil {
        log.Fatalf("could not init envconfig: %s", err)
    }
}
```

Библиотека не содержит package main

```
package greeter

func Greet(name string) string {
    return "Hello, " + name
}
```

Импортирование пакетов

```
package main

import (
    "fmt"

    "github.com/alikhanmurzayev/test_project/greeter"
)

func main() {
    fmt.Println(greeter.Greet("Alikhan"))
}
```


One-shot запуск

```
package main

import (
    "fmt"
    "github.com/alikhanmurzayev/test_project/greeter"
)

func main() {
    fmt.Println(greeter.Greet("Alikhan"))
}
```

```
$ go run main.go
```

```
alikhhan@cerebro:~/go/src/github.com/alikhanmurzayev/test_project$ go run main.go
Hello, Alikhan
```

Компиляция в исполняемый файл

\$ go build -o main.bin

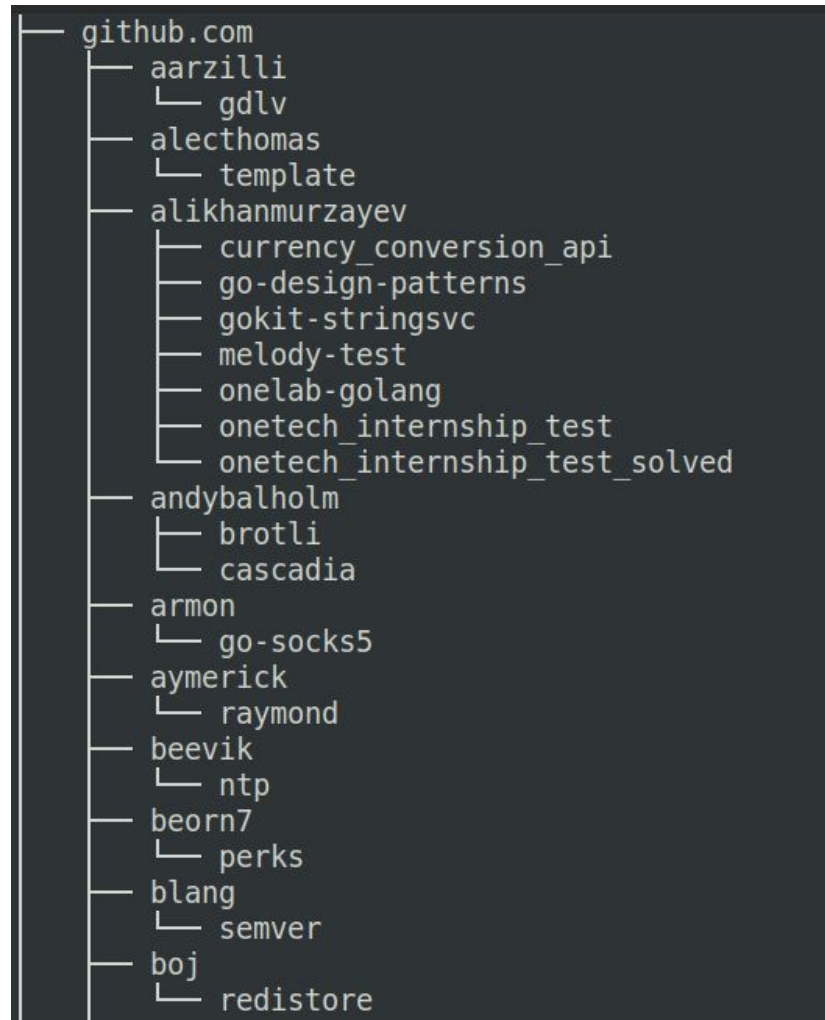
```
alikh@cerebro:~/go/src/github.com/alikhmurzayev/test_project$ go build -o main.bin
alikh@cerebro:~/go/src/github.com/alikhmurzayev/test_project$ file main.bin
main.bin: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), statically linked, Go BuildID=67PSVVP8dSjWFxNkdARE/RV6TMCyEzJVRZcWKWr3G/b00URFCJ_7cS7nAZ9u9s/PKyhWx8RpArR8IwmJkA4, not stripped
alikh@cerebro:~/go/src/github.com/alikhmurzayev/test_project$
```

Существует 2 способа организации кода

1. Единое дерево кода
2. Go modules

Единое дерево кода

\$GOPATH/go/src



Для работы с единым деревом кода нужно:

- Создать проект `github.com/username/projectname`
- Скачать проект в GOPATH с помощью `go get github.com/username/projectname/...`
- Изменять, компилировать, и комитить проект из `$GOPATH/src/github.com/username/projectname`

Преимущества:

- Простота, плоская структура

Недостатки:

- Отсутствие версий

Задание

1. Создать локально проект для библиотеки с функцией Sum:

```
func Sum(a, b int) int {  
    return a + b  
}
```

2. Создать локально проект для исполняемой программы, которая будет использовать библиотеку

Go modules

Модуль — набор go-пакетов.

В корневой директории модуля находится `go.mod` файл.

Если в директории есть `go.mod` и внутри где-то есть go-пакеты, то директория считается модулем.

Создание модуля

```
$ go mod init github.com/username/projectname
```

```
alikh@cerebro:~/Desktop/test_project$ go mod init github.com/alikhanmurzayev/test_project
go: creating new go.mod: module github.com/alikhanmurzayev/test_project
go: to add module requirements and sums:
    go mod tidy
alikh@cerebro:~/Desktop/test_project$ cat go.mod
module github.com/alikhanmurzayev/test_project

go 1.16
alikh@cerebro:~/Desktop/test_project$
```


Добавление зависимостей

```
$ go get github.com/kelseyhightower/envconfig
```

```
alikhhan@cerebro:~/Desktop/test_project$ go get github.com/kelseyhightower/envconfig
go get: added github.com/kelseyhightower/envconfig v1.4.0
alikhhan@cerebro:~/Desktop/test_project$ cat go.mod
module github.com/alikhhanmurzayev/test_project

go 1.16

require github.com/kelseyhightower/envconfig v1.4.0 // indirect
alikhhan@cerebro:~/Desktop/test_project$
```

Преимущества

- Версионирование
- Возможна работа вне GOPATH
- Защита от подмены модулей

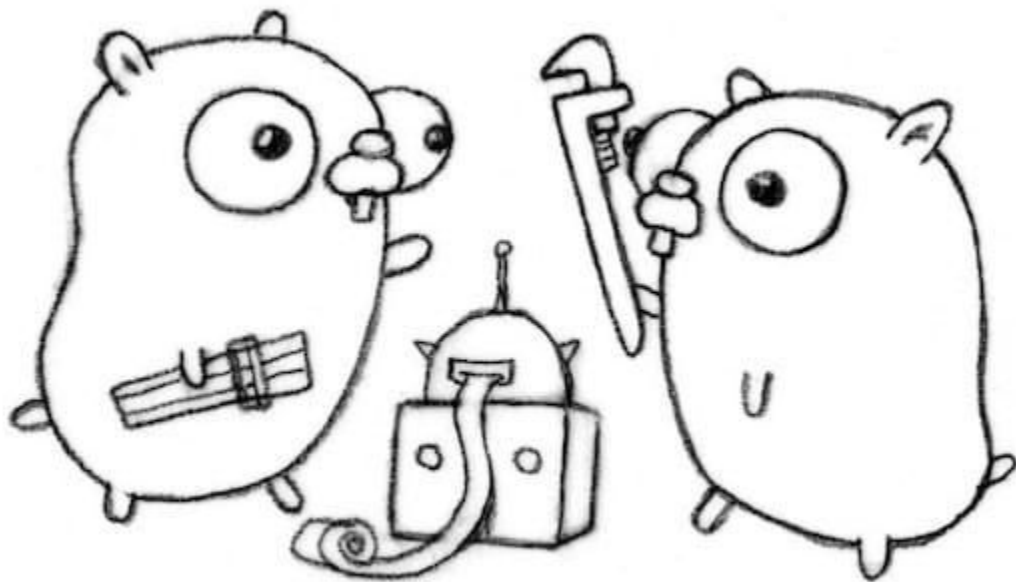
`go get` ведет себя по разному:

- При вызове в модуле скачивает в
`$GOPATH/pkg/mod`
- При вызове вне-модуля скачивает в
`$GOPATH/src`

Задание

Внедрить модули в ранее написанные программы (в библиотеку и исполняемую программу)

Утилиты



Получение справки

\$ go help

```
alikh@cerebro:~/go/src/github.com/alikhmurzayev/test_project$ go help run
usage: go run [build flags] [-exec xprog] package [arguments...]
```

Run compiles and runs the named main Go package.

Typically the package is specified as a list of .go source files from a single directory, but it may also be an import path, file system path, or pattern matching a single known package, as in 'go run .' or 'go run my/cmd'.

By default, 'go run' runs the compiled binary directly: 'a.out arguments...'.
 go run a.go

If the `-exec` flag is given, 'go run' invokes the binary using `xprog`:

```
'xprog a.out arguments...'.

```

If the `-exec` flag is not given, `GOOS` or `GOARCH` is different from the system default, and a program named `go_${GOOS}_${GOARCH}_exec` can be found

Форматирование кода:

```
$ go fmt main.go
```

```
main.go x
1 package main
2
3 import "fmt"
4
5
6 type User struct {
7     Name string `json:"name"` // is is a username
8     Phone string `json:"phone"` // this is a phone number
9     IsStudent bool `json:"is_student"`
10 }
11
12 const message = "This is a user"
13 func main() {
14     for i:=0;i<10;i++){
15         fmt.Println(i)
16         if i>=9{
17             break
18         }
19     }
20 }
21 return
22 }
23
```



```
main.go x
1 package main
2
3 import "fmt"
4
5 type User struct {
6     Name string `json:"name"` // is is a username
7     Phone string `json:"phone"` // this is a phone number
8     IsStudent bool `json:"is_student"`
9 }
10
11 const message = "This is a user"
12
13 func main() {
14     for i := 0; i < 10; i++ {
15         fmt.Println(i)
16         if i >= 9 {
17             break
18         }
19     }
20 }
21 return
22 }
23
```

Обновление импортов

```
package main

func main() {
    fmt.Println(strings.Index("hello", "l"))
}
```

→

```
package main

import (
    "fmt"
    "strings"
)

func main() {
    fmt.Println(strings.Index("hello", "l"))
}
```


Скачаем и установим утилиту ручками

1. Необходимо обновить переменную PATH в ~/.bashrc:

```
export PATH=$PATH:$GOPATH/bin
```

Скачаем и установим утилиту ручками

2. Скачаем библиотеку в \$GOPATH/src:

\$ go get -d golang.org/x/perf/cmd/benchstat

С флагом -d библиотека только скачается

```
alikhhan@cerebro:~/go/src/golang.org/x/tools/cmd$ go get -d golang.org/x/perf/cmd/benchstat
go get: added golang.org/x/perf v0.0.0-20210220033136-40a54f11e909
alikhhan@cerebro:~/go/src/golang.org/x/tools/cmd$
```

Скачаем и установим утилиту ручками

3. Установим бинарник в `$GOPATH/bin`

```
$ go install golang.org/x/perf/cmd/benchstat
```

Скачаем и установим утилиту ручками

4. Проверка

```
alikhhan@cerebro:~$ benchstat
usage: benchstat [options] old.txt [new.txt] [more.txt ...]
options:
  -alpha  $\alpha$ 
            consider change significant if  $p < \alpha$  (default 0.05)
  -csv
            print results in CSV form
  -delta-test test
            significance test to apply to delta: utest, ttest, or none (default utest)
  -geomean
            print the geometric mean of each file
  -html
            print results as an HTML table
  -norange
            suppress range columns (CSV only)
  -sort order
            sort by order: [-]delta, [-]name, none (default "none")
  -split labels
            split benchmarks by labels (default "pkg,goos,goarch")
alikhhan@cerebro:~$
```

Livecoding...



ДЗ // todo