

8. Продвинутое консольные утилиты

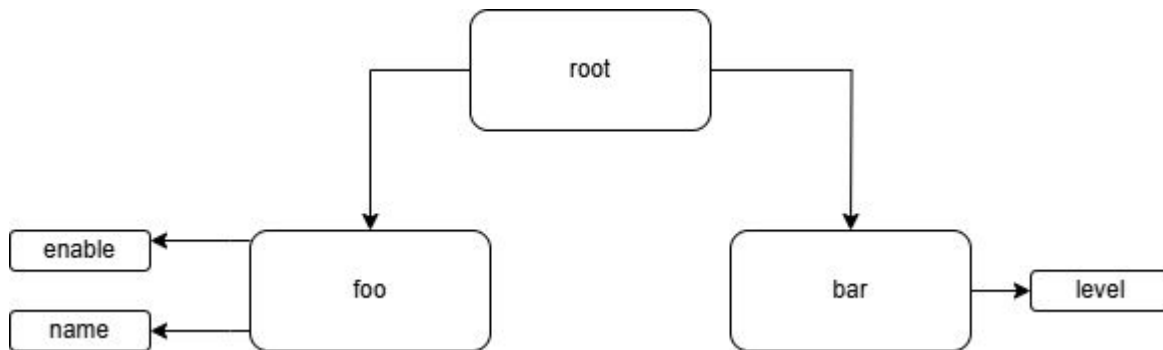
spf13/cobra

Консольные утилиты: flag

- <https://gobyexample.com/command-line-subcommands>
 - всю логику разбора команд приходится писать вручную
 - много команд – большой switch
 - приходится отсчитывать, сколько аргументов для команд парсить
 - `switch os.Args[1], fooCmd.Parse(os.Args[2:])`
 - а если хотим обязательные или необязательные флаги?
 - не наглядно!
- команды в CLI – дерево, хотелось бы:
 - автоматически подключать команду – новый узел в дереве
 - автоматически присоединять флаги к новому узлу

Консольные утилиты: flag

- команды в CLI – дерево, хотелось бы:
 - автоматически подключать команду – новый узел в дереве
 - автоматически присоединять флаги к новому узлу



spf13/cobra

- <https://github.com/spf13/cobra>
 - не стандартный пакет
- <https://pkg.go.dev/github.com/spf13/cobra>
 - документация на сайте Go
- <https://habr.com/ru/companies/otus/articles/830082/>
 - статья на русском, как использовать Cobra и viper для создания консольных утилит
- https://github.com/spf13/cobra/blob/main/site/content/user_guide.md
 - хороший официальный tutorial Cobra
- пример:
<https://github.com/katevi/golang-course/tree/master/lecture8/examples/cobra>

Добавление Cobra в проект

```
cd examples/cobra
```

```
go mod init cobra.example
```

```
go get -u github.com/spf13/cobra/cobra
```

```
1  module cobra.example
2
3  go 1.24.1
4
5  Check for upgrades | Upgrade transitive dependencies | Upgrade dire
6  require (
7      github.com/inconshreveable/mousetrap v1.1.0
8      github.com/spf13/cobra v1.9.1 // indirect
9      github.com/spf13/pflag v1.0.6 // indirect
10 )
```

```
COBRA [WSL: UBUNTU-22.04]
└─ cmd
   ├── greet.go
   ├── root.go
   └── go.mod 1
      ├── go.sum
      └── main.go
```



root.go

- <https://pkg.go.dev/github.com/spf13/cobra#Command>
- Command структуру
 - описывает команду
 - название команды
 - help
 - что надо делать, если команду вызвали
 - Execute()
 - AddCommand()
 - можно добавить флагов

```
cmd > go root.go > [e] rootCmd
1  package cmd
2
3  import (
4      "fmt"
5      "os"
6
7      "github.com/spf13/cobra"
8  )
9
10 var rootCmd = &cobra.Command{
11     Use:   "myapp",
12     Short: "A simple CLI application",
13     Long:  `A longer description of my CLI application`,
14     Run: func(cmd *cobra.Command, args []string) {
15         fmt.Println("Welcome to my app! Use --help to see commands")
16     },
17 }
18
19 func Execute() {
20     if err := rootCmd.Execute(); err != nil {
21         fmt.Println(err)
22         os.Exit(1)
23     }
24 }
```


greet.go

- команда greet:
 - подключается к команде root
 - есть флаг name

```
cmd >  greet.go >  greetCmd
1  package cmd
2
3  import (
4      "fmt"
5
6      "github.com/spf13/cobra"
7  )
8
9  var greetCmd = &cobra.Command{
10     Use:   "greet",
11     Short: "Greet someone",
12     Long:  `This command greets a person by name`,
13     Run: func(cmd *cobra.Command, args []string) {
14         name, _ := cmd.Flags().GetString("name")
15         fmt.Printf("Hello, %s!\n", name)
16     },
17 }
18
19 func init() {
20     greetCmd.Flags().StringP("name", "n", "World", "Name to greet")
21     rootCmd.AddCommand(greetCmd)
22 }
23
```


morning.go

- команда morning:
 - подключается к команде greet
 - есть флаг name

```
cmd >  morning.go >  init
1  package cmd
2
3  import (
4      "fmt"
5
6      "github.com/spf13/cobra"
7  )
8
9  var morningCmd = &cobra.Command{
10     Use:   "morning",
11     Short: "Morning greeting",
12     Run: func(cmd *cobra.Command, args []string) {
13         name, _ := cmd.Flags().GetString("name")
14         fmt.Printf("Good morning, %s!\n", name)
15     },
16 }
17
18 func init() {
19     morningCmd.Flags().StringP("name", "n", "World", "Name to greet")
20     greetCmd.AddCommand(morningCmd)
21 }
22
```

Результаты: help приложения

```
root@MSI:~/repos/my-experiments/golang-course/lecture8/examples/cobra# ./cobra.example --help
A longer description of my CLI application

Usage:
  myapp [flags]
  myapp [command]

Available Commands:
  greet      Greet someone
  help       Help about any command

Flags:
  -h, --help  help for myapp

Use "myapp [command] --help" for more information about a command.
```

Результаты: help команды greet

```
root@MSI:~/repos/my-experiments/golang-course/lecture8/examples/cobra# ./cobra.example greet --help
This command greets a person by name

Usage:
  myapp greet [flags]
  myapp greet [command]

Available Commands:
  morning      Morning greeting

Flags:
  -h, --help            help for greet
  -n, --name string     Name to greet (default "World")

Use "myapp greet [command] --help" for more information about a command.
```

Результаты: запуск приложения с разными флагами

```
root@MSI:~/repos/my-experiments/golang-course/lecture8/examples/cobra# ./cobra.example greet morning
Good morning, World!
root@MSI:~/repos/my-experiments/golang-course/lecture8/examples/cobra# ./cobra.example greet morning -n Bob
Good morning, Bob!
root@MSI:~/repos/my-experiments/golang-course/lecture8/examples/cobra# ./cobra.example greet
Hello, World!
root@MSI:~/repos/my-experiments/golang-course/lecture8/examples/cobra# ./cobra.example greet -n Kate
Hello, Kate!
```

Флаги

- бывают локальные: в пределах одной команды
- бывают персистентные
 - наследуются дочерними командами
 - подписываются как Global
- заведем флаг verbose в root команде
 - его будет видно во всех дочерних командах: greet, morning

```
10  var verbose bool
11
12  func init() {
13      rootCmd.PersistentFlags().BoolVarP(&verbose, "verbose", "v", false, "verbose output")
14  }
15
16  var rootCmd = &cobra.Command{
17      Use:  "myapp",
18      Short: "A simple CLI application",
19      CompletionOptions: cobra.CompletionOptions{
```

Работа персистентного флага

```
9  var morningCmd = &cobra.Command{
10      Use:   "morning",
11      Short: "Morning greeting",
12      Run: func(cmd *cobra.Command, args []string) {
13          name, _ := cmd.Flags().GetString("name")
14          if verbose {
15              fmt.Printf("Sending greeting to: %s\n", name)
16          }
17          fmt.Printf("Hello, %s!\n", name)
18      },
19  }
```

```
root@MSI:~/repos/my-experiments/golang-course/lecture8/examples/cobra# ./cobra.example greet morning --help
Morning greeting
```

Usage:

myapp greet morning [flags]

Flags:

-h, --help help for morning
-n, --name string Name to greet (default "World")

Global Flags:

-v, --verbose verbose output

spf13/viper

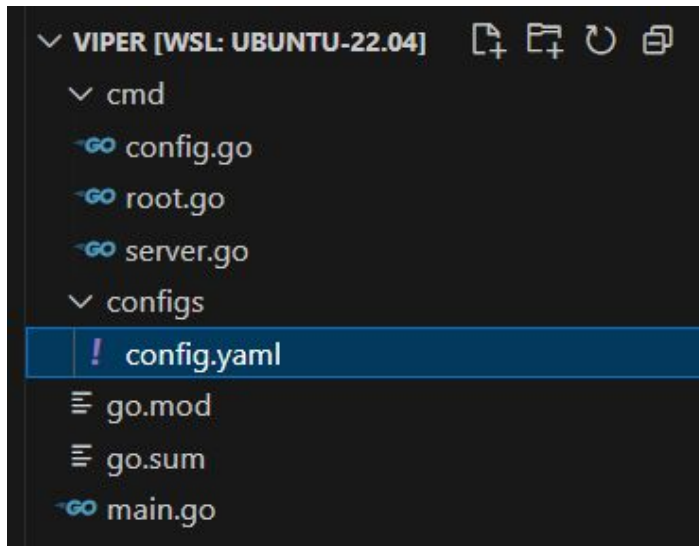
spf13/viper

- <https://github.com/spf13/viper>
 - проект на GitHub
 - прямо в README много примеров
- <https://pkg.go.dev/github.com/spf13/viper>
 - документация на сайте Go
- <https://habr.com/ru/companies/otus/articles/830082/>
 - там есть и про Viper
- пример:
<https://github.com/katevi/golang-course/tree/master/lecture8/examples/viper>

Добавление viper в проект

- `cd examples/viper`
- `go mod init viper.example`
- `go get github.com/spf13/viper`
- `go get github.com/spf13/cobra`

```
1  module viper.example
2
3  go 1.24.1
4
5  Check for upgrades | Upgrade transitive dependencies | Upgrade direct dependencies
6  require (
7      github.com/fsnotify/fsnotify v1.8.0 // indirect
8      github.com/go-viper/mapstructure/v2 v2.2.1 // indirect
9      github.com/inconshreveable/mousetrap v1.1.0 // indirect
10     github.com/pelletier/go-toml/v2 v2.2.3 // indirect
11     github.com/sagikazarmark/locafero v0.7.0 // indirect
12     github.com/sourcegraph/conc v0.3.0 // indirect
13     github.com/spf13/afero v1.12.0 // indirect
14     github.com/spf13/cast v1.7.1 // indirect
15     github.com/spf13/cobra v1.9.1 // indirect
16     github.com/spf13/pflag v1.0.6 // indirect
17     github.com/spf13/viper v1.20.1 // indirect
```



Чтение конфиг файла

- <https://pkg.go.dev/github.com/spf13/viper>
- SetConfigName – имя файла
- AddConfigPath – путь к файлам
- SetDefault – параметры с дефолтами
- AutomaticEnv() – поддержка переменных окружения

```
1 package cmd
2
3 import (
4     "fmt"
5
6     "github.com/spf13/viper"
7 )
8
9 func initConfig() {
10     // 1. Set config file name and paths
11     viper.SetConfigName("config") // Looks for config.yaml/config.json/etc
12     viper.AddConfigPath(".")       // Current directory
13     viper.AddConfigPath("./configs") // Configs subfolder
14
15     // 2. Set default values
16     viper.SetDefault("server.port", "8080")
17     viper.SetDefault("logging.level", "info")
18
19     viper.AutomaticEnv() // Read ENV variables
20     viper.SetEnvPrefix("MYAPP") // MYAPP_SERVER_PORT=5000
21
22     // Bind ENV to config keys:
23     viper.BindEnv("server.port")
24
25     // 3. Read config file
26     if err := viper.ReadInConfig(); err != nil {
27         if _, ok := err.(viper.ConfigFileNotFoundError); ok {
28             fmt.Println("No config file found - using defaults")
29         } else {
30             panic(fmt.Errorf("Fatal error in config file: %w", err))
31         }
32     }
33 }
34
```

Конфигурационный файл

```
configs > ! config.yaml
```

```
1  ∨ server:
```

```
2    port: "4444" # Overrides default 8080
```

```
3    host: "localhost"
```

```
4
```

```
5  ∨ logging:
```

```
6    level: "debug" # Overrides default "info"
```

```
7
```

root.go

- вызываем функцию чтения конфига

```
cmd > -go root.go > ...
1  package cmd
2
3  import (
4      "fmt"
5      "os"
6
7      "github.com/spf13/cobra"
8  )
9
10 var rootCmd = &cobra.Command{
11     Use:   "myapp",
12     Short: "A Viper-powered app",
13     CompletionOptions: cobra.CompletionOptions{
14         DisableDefaultCmd: true,
15         DisableDescriptions: true,
16         DisableNoDescFlag: true,
17     },
18     PersistentPreRun: func(cmd *cobra.Command, args []string) {
19         initConfig() // Initialize Viper before any command runs
20     },
21 }
22
23 func Execute() {
24     if err := rootCmd.Execute(); err != nil {
25         fmt.Println(err)
26         os.Exit(1)
27     }
28 }
```

server.go

- viper.GetString() – получить значение строкового параметра из конфигурационного файла по имени
- server.port, logging.level
- viper.BindPFlag() – перезаписать значение параметра server.port значением флага

```
cmd > server.go > serverCmd
1  package cmd
2
3  import (
4      "fmt"
5
6      "github.com/spf13/cobra"
7      "github.com/spf13/viper"
8  )
9
10 func init() {
11     serverCmd.Flags().StringP("port", "p", "", "Server port")
12     viper.BindPFlag("server.port", serverCmd.Flags().Lookup("port"))
13
14     // Command-line will override config file:
15     // ./myapp start -p 4000
16 }
17
18 // Example command using Viper values
19 var serverCmd = &cobra.Command{
20     Use:   "start",
21     Short: "Start the server",
22     Run: func(cmd *cobra.Command, args []string) {
23         port := viper.GetString("server.port")
24         logLevel := viper.GetString("logging.level")
25
26         fmt.Printf("Starting server on %s with log level %s\n",
27             port, logLevel)
28     },
29 }
30
31 func init() {
32     rootCmd.AddCommand(serverCmd)
33 }
34
```

Результаты запуска приложения

- указываем порт через флаг (1111)
 - перезаписывается значение из конфига и дефолт
- не указываем порт через флаг
 - берется значение из конфига (4444)
- не указываем порт в конфиге
 - берется дефолтное значение (8080)

```
root@MSI:~/repos/my-experiments/golang-course/lecture8/examples/viper# ./viper.example start --port 1111
Starting server on 1111 with log level debug
root@MSI:~/repos/my-experiments/golang-course/lecture8/examples/viper# ./viper.example start
Starting server on 4444 with log level debug
root@MSI:~/repos/my-experiments/golang-course/lecture8/examples/viper# ./viper.example start
Starting server on 8080 with log level debug
```