文档

github地址: https://github.com/spf13/viper

https://www.liwenzhou.com/posts/Go/viper_tutorial/

项目实例

```go
func Viper(path ...string) *viper.Viper {
    var config string
    if len(path) == 0 {
        flag.StringVar(&config, "c", "", "choose config file.")
        flag.Parse()
        if config == "" { // 优先级: 命令行 > 环境变量 > 默认值
            if configEnv := os.Getenv(utils.ConfigEnv); configEnv == "" {
                config = utils.ConfigFile
                fmt.Printf("您正在使用config的默认值,config的路径为%v\n", utils.ConfigFile)
            } else {
                config = configEnv
                fmt.Printf("您正在使用GVA_CONFIG环境变量,config的路径为%v\n", config)
            }
        } else {
            fmt.Printf("您正在使用命令行的-c参数传递的值,config的路径为%v\n", config)
        }
    } else {
        config = path[0]
        fmt.Printf("您正在使用func Viper()传递的值,config的路径为%v\n", config)
    }

    v := viper.New()
    v.SetConfigFile(config) // 设置配置文件
    v.SetConfigType("yaml")    // 配置文件格式
    err := v.ReadInConfig()
    if err != nil {
        panic(fmt.Errorf("Fatal error config file: %s \n", err))
    }

```

```
    // 开启配置文件的更新
    v.WatchConfig()
    v.OnConfigChange(func(e fsnotify.Event) {
        fmt.Println("config file changed:", e.Name)
        if err := v.Unmarshal(&global.GVA_CONFIG); err != nil {    // 回调动作
            fmt.Println(err)
        }
    })
    if err := v.Unmarshal(&global.GVA_CONFIG); err != nil {
        fmt.Println(err)
    }
    // root 适配性
    // 根据root位置去找到对应迁移位置,保证root路径有效
    global.GVA_CONFIG.AutoCode.Root, _ = filepath.Abs("..")
    global.BlackCache = local_cache.NewCache(
        local_cache.SetDefaultExpire(time.Second *
time.Duration(global.GVA_CONFIG.JWT.ExpiresTime)),
    )
    return v
}

```

v.Unmarshal(&gobal.GVA_CONFIG)

反序列化对象,需要使用mapstructure tag标签, 可以反序列化到嵌套结构

```
package config

type Server struct {
    JWT     JWT     `mapstructure:"jwt" json:"jwt" yaml:"jwt"`
    Zap     Zap     `mapstructure:"zap" json:"zap" yaml:"zap"`
    Redis   Redis   `mapstructure:"redis" json:"redis" yaml:"redis"`
    Email   Email   `mapstructure:"email" json:"email" yaml:"email"`
    Casbin  Casbin  `mapstructure:"casbin" json:"casbin" yaml:"casbin"`
    System  System  `mapstructure:"system" json:"system" yaml:"system"`
    Captcha Captcha `mapstructure:"captcha" json:"captcha" yaml:"captcha"`
    // auto
    AutoCode Autocode `mapstructure:"autoCode" json:"autoCode" yaml:"autoCode"`
    // gorm
    Mysql  Mysql `mapstructure:"mysql" json:"mysql" yaml:"mysql"`
    Pgsql  Pgsql `mapstructure:"pgsql" json:"pgsql" yaml:"pgsql"`
```

```
    DBList []DB  `mapstructure:"db-list" json:"db-list" yaml:"db-list"`
    // oss
    Local      Local      `mapstructure:"local" json:"local" yaml:"local"`
    Qiniu      Qiniu      `mapstructure:"qiniu" json:"qiniu" yaml:"qiniu"`
    AliyunOSS  AliyunOSS  `mapstructure:"aliyun-oss" json:"aliyunOSS" yaml:"aliyun-oss"`
    HuaWeiObs  HuaWeiObs  `mapstructure:"hua-wei-obs" json:"huaWeiObs" yaml:"hua-wei-obs"`
    TencentCOS TencentCOS `mapstructure:"tencent-cos" json:"tencentCOS" yaml:"tencent-cos"`
    AwsS3      AwsS3      `mapstructure:"aws-s3" json:"awsS3" yaml:"aws-s3"`

    Excel Excel `mapstructure:"excel" json:"excel" yaml:"excel"`
    Timer Timer `mapstructure:"timer" json:"timer" yaml:"timer"`

    // 跨域配置
    Cors CORS `mapstructure:"cors" json:"cors" yaml:"cors"`
}

```