链滴

Golang 实现默认参数

作者：Allenxuxu

原文链接：https://ld246.com/article/1561595948850

来源网站：链滴

在golang 中是不支持默认参数的，micro中有一种优雅的实现方法(并非 micro 首创)，叫做 Functional Options Patter。Functional Options 可以用来实现简洁的支持默认参数的函数方法。

options

```
package server


import (
    "time"
)

type Options struct {
    ConnectTimeOut time.Duration
    Name           string
    Address        string
}

type  Option  func(*Options)

func newOptions(opt ...Option) Options {
    opts := Options{}

    for _, o := range opt {
        o(&opts)
    }

    if len(opts.Address) == 0 {
        opts.Address = DefaultAddress
    }

    if len(opts.Name) == 0 {
        opts.Name = DefaultName
    }

    if opts.ConnectTimeOut == time.Duration(0) {
        opts.ConnectTimeOut = DefaultConnectTimeOut
    }

    return opts
}

// Name server name
func Name(n string) Option {
    return func(o *Options) {
        o.Name = n
    }
}

// Address server address
func Address(a string) Option {
    return func(o *Options) {
        o.Address = a
```

原文链接：Golang 实现默认参数

```
    }
}

// ConnectTimeOut 连接超时时间
func ConnectTimeOut(t time.Duration) Option {
    return func(o *Options) {
        o.ConnectTimeOut = t
    }
}
```

server

```
package server

import "sync"

var (
    DefaultAddress = ":0"
    DefaultName = "server"
    DefaultConnectTimeOut = time.Second * 4
)

type Server struct {
    sync.RWMutex
    opts Options
}

func NewServer(opts ...Option) Server {
    options := newOptions(opts...)
    return &Server{
        opts: options,
    }
}

func (s *Server) Options() Options {
    s.RLock()
    opts := s.opts
    s.RUnlock()
    return opts
}

func (s *Server) Init(opts ...Option) error {
    s.Lock()
    for _, opt := range opts {
        opt(&s.opts)
    }
    s.Unlock()
    return nil
}

func (s *Server) Start() error {
    return nil
}
```

```
func (s *Server) Stop() error {
    return  nil
}
```

使用

```
server := NewServer(
    Name("test Name"),
    Address("test Address"),
 )
```