Go programming patterns

Dec 29, 2020

罗健文

Overview

- GopherChina 2020
- Go programming patterns

GopherChina 2020

3

GopherChina 2020

1.2 探探分布式存储的实践(gocn.vip).pdf	gopher china 2020 ppt
1.3 go-programming-patterns(gocn.vip).pdf	gopher china 2020 ppt
1.4 Golang in GrabFood Discovery System(gocn.vip).pdf	gopher china 2020 ppt
1.5 华为云的go语言云原生实战经验(gocn.vip).pdf	gopher china 2020 ppt
1.6 Functional options and config for APIs(gocn.vip).pdf	gopher china 2020 ppt
2.1.1 百度万亿流量转发平台背后的故事(gocn.vip).pdf	gopher china 2020 ppt
2.1.2 Go如何帮滴滴支撑海量运维场景(gocn.vip).pdf	gopher china 2020 ppt
2.1.3 如何用go module构建模块化跨链平台(gocn.vip).pdf	gopher china 2020 ppt
2.1.5 Go语言编译器简介(gocn.vip).pdf	gopher china 2020 ppt
2.1.6 基于TarsGo的云原生微服务架构演进(gocn.vip).pdf	gopher china 2020 ppt
2.1.7 使用Golang实现万人同服的游戏服务器(gocn.vip).pdf	gopher china 2020 ppt
2.2.1 Go in the Cloud - Why People Choose Go for Cloud...	gopher china 2020 ppt
2.2.2 云原生go-zero微服务框架设计思考(gocn.vip).pdf	gopher china 2020 ppt
2.2.3 Go+演进之路(gocn.vip).pdf	gopher china 2020 ppt
2.2.4 直播长链接架构演进(gocn.vip).pdf	gopher china 2020 ppt
2.2.5 go-Chassis 在 shopee 供应链的实践(gocn.vip).pdf	gopher china 2020 ppt
2.2.6 Golang大规模云原生应用管理实践(gocn.vip).pdf	gopher china 2020 ppt
2.2.7 GORM 剖析与最佳实践(gocn.vip).pdf	gopher china 2020 ppt

Go programming patterns

5

Go programming patterns

The Creators of Golang

Robert Griesemer	Rob Pike	Ken Thompson
Born 1964	Born 1956	Born 1943
Java HotSpot JVM	Unix	Unix/C
V8 Javascript engine	UTF-8	UTF-8
Golang	Golang	Golang

6

Go programming patterns

7

Topics

- Functional Options
- Error Handing

8

Functional Options

你有一个任务就是编写一些关键的服务组件，就像这样的代码：

```
page gplusplus

import "net"

type Server struct {
    listener net.Listener
}

func (s *Server) Add() net.Addr
func (s *Server) Shutdown()

// NewServer returns a new Server listenering on addr.
func NewServer(addr string) (*Server, error) {
    l, err := net.Listen("tcp", addr)
    if err !=nil {
        return nil, err
    }

    srv := Server{listener: l}
    go src.run()
    return &srv, nil
}
```

9

Functional Options

新需求来了

- does your server support TLS ?
- can I limit the number of clients ?
- can I specify the port ?
- can I set the timeout ?
- ... and more ...

```
func NewServer(addr string, port int, maxconns int, timeout time.Duration, tls *tls.Config) (*Server, er
    //...
}
```

10

Functional Options

```
// We need a number of New Server function for different senarios
func NewServer(addr string, port int) (*Server, error) {
    //...
}
func NewTLSServer(addr string, port int, tls *tls.Config) (*Server, error) {
    //...
}
func NewServerWithTimeout(addr string, port int, timeout time.Duration) (*Server, error) {
    //...
}
func NewTLSServerWithMaxConnAndTimeout(addr string, port int, maxconns int, timeout time.Duration, tls *
    //...
}
```

11

One Popular Solution

```
type Config struct {
    Protocol string
    Timeout  time.Duration
    Maxconns int
    TLS      *tls.Config
}

func NewServer(add string, port int, config *Config) (*Server, error) {
    //...
}

srv1, _ := NewServer("localhost", 9000, nil)
```

```
conf := Config{Protocol: "tcp", Timeout: 60 * time.Second, Maxconns: 10}
svc2, _ := NewServer("localhost", 9000, &conf)
```

```
conf.Maxconns=15
```

12

Functional Options

```
func NewServer(add string, port int, config ...*Config) (*Server, error) {
    //...
}

func main() {
    svc1, _ := NewServer("localhost", 9000)
    svc2, _ := NewServer("localhost", 9000, &Config{
        Timeout:  10 * time.Second,
        Maxconns: 10,
    })
}
```

13

Functional Options

```
type Option func(*Server)

func Protocol(p string) Option {
    return func(s *Server) {
        s.Protocol = p
    }
}

func Timeout(timeout time.Duration) Option {
    return func(s *Server) {
        s.Timeout = timeout
    }
}

func MaxConns(maxconns int) Option {
    return func(s *Server) {
        s.Maxconns = maxconns
    }
}

func TLS(tls *tls.Config) Option {
    return func(s *Server) {
        s.TLS = tls
    }
}
```

Functional Options

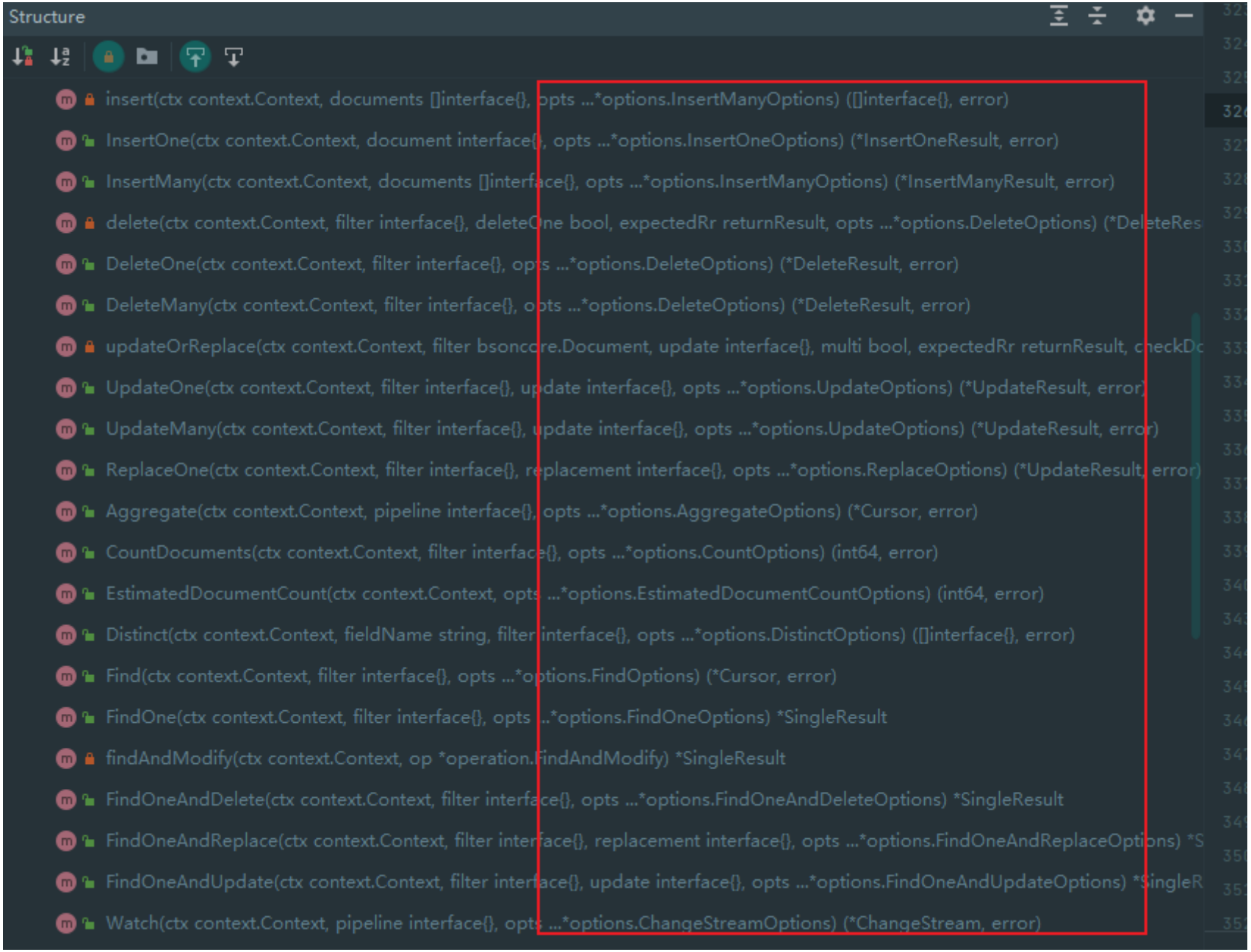
```
func NewServer(addr string, port int, options ...func(*Server)) (*Server, error) {
    l, err := net.Listen("tcp", addr)
    if err !=nil {
        return nil, err
    }

    srv := Server{listener: l}
    for _, option := range options {
        option(&srv)
    }
    return &srv, nil
}
```

Article:

self-referential-functions-and-design--Robe Pike (https://commandcenter.blogspot.com/2014/01/self-referential-functions-and-design.html)

functional-options-for-friendly-apis--Dave Cheney (https://dave.cheney.net/2014/10/17/functional-options-for-friendly-apis) 15

Functional Options


Structure
insert(ctx context.Context, documents []interface{}, opts ...*options.InsertManyOptions) ([]interface{}, error)
InsertOne(ctx context.Context, document interface{}, opts ...*options.InsertOneOptions) (*InsertOneResult, error)
InsertMany(ctx context.Context, documents []interface{}, opts ...*options.InsertManyOptions) (*InsertManyResult, error)
delete(ctx context.Context, filter interface{}, deleteOne bool, expectedRr returnResult, opts ...*options.DeleteOptions) (*DeleteRes
DeleteOne(ctx context.Context, filter interface{}, opts ...*options.DeleteOptions) (*DeleteResult, error)
DeleteMany(ctx context.Context, filter interface{}, opts ...*options.DeleteOptions) (*DeleteResult, error)
updateOrReplace(ctx context.Context, filter bsoncore.Document, update interface{}, multi bool, expectedRr returnResult, checkDo
UpdateOne(ctx context.Context, filter interface{}, update interface{}, opts ...*options.UpdateOptions) (*UpdateResult, error)
UpdateMany(ctx context.Context, filter interface{}, update interface{}, opts ...*options.UpdateOptions) (*UpdateResult, error)
ReplaceOne(ctx context.Context, filter interface{}, replacement interface{}, opts ...*options.ReplaceOptions) (*UpdateResult, error)
Aggregate(ctx context.Context, pipeline interface{}, opts ...*options.AggregateOptions) (*Cursor, error)
CountDocuments(ctx context.Context, filter interface{}, opts ...*options.CountOptions) (int64, error)
EstimatedDocumentCount(ctx context.Context, opts ...*options.EstimatedDocumentCountOptions) (int64, error)
Distinct(ctx context.Context, fieldName string, filter interface{}, opts ...*options.DistinctOptions) ([]interface{}, error)
Find(ctx context.Context, filter interface{}, opts ...*options.FindOptions) (*Cursor, error)
FindOne(ctx context.Context, filter interface{}, opts ...*options.FindOneOptions) *SingleResult
findAndModify(ctx context.Context, op *operation.FindAndModify) *SingleResult
FindOneAndDelete(ctx context.Context, filter interface{}, opts ...*options.FindOneAndDeleteOptions) *SingleResult
FindOneAndReplace(ctx context.Context, filter interface{}, replacement interface{}, opts ...*options.FindOneAndReplaceOptions) *S
FindOneAndUpdate(ctx context.Context, filter interface{}, update interface{}, opts ...*options.FindOneAndUpdateOptions) *SingleR
Watch(ctx context.Context, pipeline interface{}, opts ...*options.ChangeStreamOptions) (*ChangeStream, error)

16

Error Handing

Go 语言的函数支持多返回值，所以，可以在返回接口把业务语义（业务返回值）和控制语义（出错返回值）区分开来。Go 语言的很多函数都会返回 result, err 两个值。

- 参数上基本上就是入参，而返回接口把结果和错误分离，这样使得函数的接口语义清晰；
- 而且，Go 语言中的错误参数如果要忽略，需要显式地忽略，用 _ 这样的变量来忽略；
- 另外，因为返回的 error 是个接口（其中只有一个方法 Error()，返回一个 string ），所以你可以扩展自定义的错误处理。

17

Error Handing

如果一个函数返回了多个不同类型的 error，可以这样做：

```
if err != nil {
switch err.(type) {
    case *json.SyntaxError:
    ...
    case *ZeroDivisionError:
    ...
    case *NullPointerError:
    ...
    default:
    ...
}
}
```

18

Error Check Hell

```
func parse(r io.Reader) (*Point, error) {

    var p Point

    if err := binary.Read(r, binary.BigEndian, &p.Longitude); err != nil {
        return nil, err
    }
    if err := binary.Read(r, binary.BigEndian, &p.Latitude); err != nil {
        return nil, err
    }
    if err := binary.Read(r, binary.BigEndian, &p.Distance); err != nil {
        return nil, err
    }
    if err := binary.Read(r, binary.BigEndian, &p.ElevationGain); err != nil {
        return nil, err
    }
    if err := binary.Read(r, binary.BigEndian, &p.ElevationLoss); err != nil {
        return nil, err
    }
}
```

19

Error Handing

```
func parse(r io.Reader) (*Point, error) {
    var p Point
    var err error
    read := func(data interface{}) {
        if err != nil {
            return
        }
        err = binary.Read(r, binary.BigEndian, data)
    }

    read(&p.Longitude)
    read(&p.Latitude)
    read(&p.Distance)
    read(&p.ElevationGain)
    read(&p.ElevationLoss)

    if err != nil {
        return &p, err
    }
    return &p, nil
}
```

Error Handing

可以通过函数式编程的方式

```
func parse(r io.Reader) (*Point, error) {
    var p Point
    var err error
    read := func(data interface{}) {
        if err != nil {
            return
        }
        err = binary.Read(r, binary.BigEndian, data)
    }

    read(&p.Longitude)
    read(&p.Latitude)
    read(&p.Distance)
    read(&p.ElevationGain)
    read(&p.ElevationLoss)

    if err != nil {
        return &p, err
    }
    return &p, nil
}
```

21

Error Handing

```
type Reader struct {
    r   io.Reader
    err error
}

func (r *Reader) read(data interface{}) {
    if r.err == nil {
        r.err = binary.Read(r.r, binary.BigEndian, data)
    }
}

func parse(input io.Reader) (*Point, error) {
    var p Point
    r := Reader{r: input}

    r.read(&p.Longitude)
    r.read(&p.Latitude)
    r.read(&p.Distance)
    r.read(&p.ElevationGain)
    r.read(&p.ElevationLoss)
    if r.err != nil {
        return nil, r.err
    }
    return &p, nil
}
```

Error Handing

Article:

golang-error-handling-lesson-by-rob-pike (http://jxck.hatenablog.com/entry/golang-error-handling-lesson-by-rob-pike)

Go编程模式——错误处理--左耳耗子 (https://coolshell.cn/articles/21140.html)

errors-are-values (https://blog.golang.org/errors-are-values)

资料

Golang中present工具--萝卜头LJW (https://blog.csdn.net/u013164931/article/details/101004573)

GopherChina2020-ppt (https://github.com/robotLJW/GopherChina)

左耳耗子的博客 (https://coolshell.cn/)

24

Thank you

罗健文