Context

Como usar?
17 November 2017

Felipe Oliveira
Senior Software Engineer, Nuveo

Sobre mim

Felipe Oliveira @_felipeweb (https://www.felipeweb.net.br)

- Mais de 3 anos de expriência com Go

Projetos

- Nuveo Software
- pREST (http://postgres.rest)
- gofn (https://github.com/nuveo/gofn)
- mercurius (https://github.com/novatrixtech/mercurius)

Nuveo

Logo Nuveo (https://www.nuveo.ai)

- Mais ou menos 90% da base de Código em Go
- Clientes como Bradesco, Cyrela, Vivo e outros grandes

Para que serve o pacote context?

- Deadline
- Timeout
- Cancelamento de sinais
- Escopo de requisição

Para que serve o pacote context?

```
type Context interface {
    // Done returns a channel that is closed when this Context is canceled
    // or times out.
    Done() <-chan struct{}

    // Err indicates why this context was canceled, after the Done channel
    // is closed.
    Err() error

    // Deadline returns the time when this Context will be canceled, if any.
    Deadline() (deadline time.Time, ok bool)

    // Value returns the value associated with key or nil if none.
    Value(key interface{}) interface{}
}
```

Context Background

```
// Background returns an empty Context. It is never canceled, has no deadline,
// and has no values. Background is typically used in main, init, and tests,
// and as the top-level Context for incoming requests.
func Background() Context {}
```

- Raiz da arvore de contextos
- Não tem `CancelFunc`
- Não tem deadline

Context WithCancel

```
// WithCancel returns a copy of parent whose Done channel is closed as soon as
// parent.Done is closed or cancel is called.
func WithCancel(parent Context) (ctx Context, cancel CancelFunc) {}

// A CancelFunc cancels a Context.
type CancelFunc func()
```

- Contexto pai
- Execução da `CancelFunc`
- Não tem deadline

Context WithDeadline

```
// WithDeadline returns a copy of the parent context with the deadline adjusted
// to be no later than d. If the parent's deadline is already earlier than d,
// WithDeadline(parent, d) is semantically equivalent to parent. The returned
// context's Done channel is closed when the deadline expires, when the returned
// cancel function is called, or when the parent context's Done channel is
// closed, whichever happens first.
//
// Canceling this context releases resources associated with it, so code should
// call cancel as soon as the operations running in this Context complete.
func WithDeadline(parent Context, d time.Time) (Context, CancelFunc) {}
```

- Contexto pai
- Execução da `CancelFunc`
- Utrapassar o tempo limite

Context WithTimeout

```
// WithTimeout returns WithDeadline(parent, time.Now().Add(timeout)).
//
// Canceling this context releases resources associated with it, so code should
// call cancel as soon as the operations running in this Context complete
func WithTimeout(parent Context, timeout time.Duration) (Context, CancelFunc) {
    return WithDeadline(parent, time.Now().Add(timeout))
}
```

- WithTimeout == WithDeadline(now + duration)

Context WithValue

```
// WithValue returns a copy of parent in which the value associated with key is
// val.
//
// Use context Values only for request-scoped data that transits processes and
// APIs, not for passing optional parameters to functions.
//
// The provided key must be comparable and should not be of type
// string or any other built-in type to avoid collisions between
// packages using context. Users of WithValue should define their own
// types for keys. To avoid allocating when assigning to an
// interface{}, context keys often have concrete type
// struct{}. Alternatively, exported context key variables' static
// type should be a pointer or interface.
func WithValue(parent Context, key, val interface{}) Context {}
```

- Contexto pai
- Não tem `CancelFunc`
- Não tem deadline

Exemplo de uso context.WithValue

```
func FromContext(ctx context.Context) (net.IP, bool) {
    // ctx.Value returns nil if ctx has no value for the key;
    // the net.IP type assertion returns ok=false for nil.
    userIP, ok := ctx.Value(userIPKey).(net.IP)
    return userIP, ok
}
```

Exemplo de uso outros contextos

```
func Run(ctx context.Context, buildOpts *provision.BuildOptions, containerOpts *provision.ContainerOptio
    var client *docker.Client
    var container *docker.Container
    var machine *iaas.Machine
    done := make(chan struct{})
    go func(ctx context.Context, done chan struct{}) {
        // omited code
        done <- struct{}{}
    }(ctx, done)
    select {
    case <-ctx.Done():
        log.Printf("trying to destroy container %v\n", ctx.Err())
    case <-done:
        log.Println("trying to destroy container process done")
    }
    // omited code
    return
} // fim
```

Mais exemplos

```
func queueLoop(conn *beanstalk.Conn) {
    // omited code
    // ajusta o status do job
    ctx, cancelFunc := context.WithTimeout(context.Background(), item.Timeout)
    // omited code
    if err = sc.Err(); err != nil {
        decrementJobCount(customer)
        log.Errorln(err)
        cancelFunc()
        continue
    }
    // executa o runner

    go func(ctx context.Context, cancelFunc context.CancelFunc) {
        err = runFn(ctx, runJob, item, id, conn)
        if err != nil {
            log.Errorln(err)
        }
        decrementJobCount(customer)
        cancelFunc()
    }(ctx, cancelFunc, customer, item, id, conn)
} // fim
```

Armadilhas no uso context

- Aumento do uso de memória da aplicação
- Se o contexto pai for contexto do tipo deadline, mantem o tempo do contexto pai

Thank you

Felipe Oliveira
Senior Software Engineer, Nuveo
felipeweb.programador@gmail.com (mailto:felipeweb.programador@gmail.com)
https://www.felipeweb.net.br (https://www.felipeweb.net.br)
@_felipeweb (http://twitter.com/_felipeweb)