Урок Спринт 2.14

Контексты и отмена с таймаутом

В предыдущих уроках мы рассматривали пакет `context`. Вспомним функцию `WithTimeout` из этого пакета.

```
func WithTimeout(parent Context, timeout time.Duration) (Context, CancelFunc)
```

`WithTimeout` возвращает `WithDeadline(parent, time.Now().Add(timeout))`. Отмена этого контекста освобождает связанные с ним ресурсы, поэтому код должен вызвать отмену, как только операции, выполняемые в этом контексте, завершатся.

```
func slowOperationWithTimeout(ctx context.Context) (Result, error) {
    ctx, cancel := context.WithTimeout(ctx, 100*time.Millisecond)
    defer cancel() // освобождает ресурсы, если slowOperation завершается до истечения таймаута
    return slowOperation(ctx)
}
```

Рассмотрим пример с использованием контекста с таймаутом:

```
package main

import (
    "context"
    "fmt"
    "time"
)

func main() {
    //создание контекста WithTimeout, который ограничивает продолжительность в течение 2 секунд
    ctx, cancel := context.WithTimeout(context.Background(), 2*time.Second)
    defer cancel()
    //создание канала chan_c
    chan_c := make(chan string)
    //создание потока выполнения горутины
    go func() {
        // создание фиктивной операции, которая занимает 3 секунды с использованием Sleep()
        time.Sleep(3 * time.Second)
        chan_c <- "Инструкции успешно завершены."
    }()
```

```
    select {
    case result := <-chan_c:
        fmt.Println(result)
    case <-ctx.Done():
        fmt.Println("Время операции истекло или было отменено.")
    }
}
```

Справка