Урок Спринт 2.13

Таймеры

В этом уроке мы познакомимся с таймерами и таймаутами, предоставляемых пакетом time стандартной библиотеки go.

Go предлагает множество функций для упрощения разработки конкурентных и сетевых программ. Таймеры — одна из таких функций, которая позволяет запланировать выполнение задачи на определенное время. Структура `Timer` пакета `time` предоставляет функциональность для выполнения задач в указанный момент времени. Таймер не начинает работать автоматически сразу после создания. Вместо этого, он ожидает определённого момента времени и запускается только однажды. Важно помнить, что таймер срабатывает один раз и не может быть использован повторно. Рассмотрим структуру `Timer` пакета `time`. Она выглядит следующим образом:

```
// Тип таймер представляет одно событие.
// По истечении таймера текущее время будет отправлено в канал C, если только таймер не был созд
// Таймер должен быть создан с помощью NewTimer или AfterFunc.
type Timer struct {
    C <-chan Time
    r runtimeTimer
}
```

`Timer` представляет собой одно событие. Когда таймер истекает, текущее время будет отправлено в канал `C`.

Рассмотрим простой пример использования таймера.

```
package main

import (
    "fmt"
    "time"
)

func main() {
    // Создание таймера, который истечет через 3 секунды.
```

```
    timer := time.NewTimer(3 * time.Second)
    // канал C отправляет значение, указывающее на истечение таймера
    <-timer.C
    fmt.Println("Time's up!")
}
```

В приведённом выше фрагменте, кода мы создали новый таймер, который истечёт через 3 секунды. Функция `main` ждёт, пока не получит значение из канала `C` таймера, указывающее на истечение таймера.

Возникает закономерный вопрос, почему мы должны использовать таймеры, когда можем просто ждать с использованием функции `time.Sleep()`. Причина заключается в том, что таймеры могут быть остановлены в любое время до срабатывания. Рассмотрим пример, чтобы увидеть, как можно остановить таймер перед его срабатыванием. Мы расширим предыдущий пример.

```
package main

import (
    "fmt"
    "time"
)

func timersStoppage(v time.Timer) {
    <-v.C
    fmt.Println("Второй таймер сработал")
}
func main() {
    // создание первого таймера
    timer := time.NewTimer(3 * time.Second)
    // в канал C отправляет значение, указывающее на истечение таймера
    <-timer.C
    fmt.Println("Первый таймер сработал!")
    // создание второго таймера
    timer_s := time.NewTimer(time.Second)
    // создание горутины
    go timersStoppage(*timer_s)
    // удалите комментарий из строки ниже, чтобы сработал второй таймер
    // timersStoppage((*timer_s))
    // остановка второго таймера перед срабатыванием
    stop_s := timer_s.Stop()
    if stop_s {
        fmt.Println("Второй таймер остановлен")
    }
}
```

В приведенном выше примере у второго таймера есть одна секунда на срабатывание, но мы остановили его, прежде чем он получил возможность сработать. Если вы хотите увидеть, как сработал второй таймер, удалите комментарий `//timersStoppage((*timer_s))`. Вы увидите, что второй таймер тоже сработал.

Справка