Урок Спринт 1.12

Примеры

Пример 1: Определение продолжительности выполнения операции

Измеряем время выполнения операции и обрабатываем ошибку, если она выполняется слишком долго:

```
package main

import (
    "fmt"
    "time"
)

func main() {
    // Записываем текущее время до начала операции
    startTime := time.Now()

    // Здесь вы можете выполнять какую-нибудь долгую операцию —
    // например, обработку большого объема данных или вычисления

    // Допустим, мы выполняем операцию, которая имитирует задержку
    for i := 0; i < 1000000; i++ {
        // Делаем что-то
    }

    // Записываем время после того, как операция выполнена
    elapsed := time.Since(startTime)

    // Проверяем, сколько времени заняла операция
    // Если операция заняла более 1 секунды, мы считаем её слишком долгой
    if elapsed.Seconds() > 1 {
        fmt.Println("Операция заняла слишком много времени")
    } else {
        fmt.Println("Операция завершена за", elapsed)
    }
}
```

Здесь мы используем `time.Now()`, чтобы получить временную метку перед началом операции и ещё одну, после того, как операция выполнена. Затем вычисляем разницу между двумя этими моментами времени с помощью `time.Since(startTime)`. Если операция заняла больше 1 секунды, мы выводим сообщение о том, что она выполняется слишком долго.

Пример 2: Планирование выполнения задачи с использованием таймера

Используем таймер, чтобы запланировать выполнение задачи через определённое время:

```
package main

import (
    "fmt"
    "time"
)

func main() {
    // Устанавливаем интервал времени для таймера (в данном случае, 5 секунд)
    interval := 5 * time.Second

    // Создаём новый таймер с указанным интервалом
    timer := time.NewTimer(interval)

    fmt.Println("Задача будет выполнена через", interval)
```

```
    // Ожидаем события от таймера (пока не прошло 5 секунд)
    <-timer.C

    fmt.Println("Задача выполнена!")
}
```

Здесь мы используем оператор `<-timer.C`. О том, зачем он нужен, мы поговорим в одном из следующих занятий.

Мы создаём таймер с интервалом в 5 секунд, и затем ждем, пока он истечёт. Как только таймер сработает, на экране появится сообщение «Задача выполнена!».

Пример 3: Работа с часовыми поясами и форматирование времени

Устанавливаем часовой пояс и форматируем время в нужный формат:

```
package main

import (
    "fmt"
    "time"
)

func main() {
    // Устанавливаем часовой пояс UTC
    loc, err := time.LoadLocation("UTC")
    if err != nil {
        fmt.Println("Ошибка при загрузке часового пояса:", err)
        return
    }

    // Получаем текущее время в указанном часовом поясе
    currentTime := time.Now().In(loc)

    // Форматируем время в строку с заданным форматом
    formattedTime := currentTime.Format("2006-01-02 15:04:05")

    fmt.Println("Текущее время (UTC):", formattedTime)
}
```

Здесь мы устанавливаем часовой пояс UTC с помощью `time.LoadLocation`, получаем текущее время с учетом этого часового пояса и форматируем его в строку с помощью метода `Format`. Наш результат — время в формате "2006-01-02 15:04:05".

Теперь вы знаете, как управлять временем в Go. Попробуйте применить новые знания на практике!