09

Testando múltiplas vezes

Transcrição

Queremos rodar o código que testa o site para cada um dos sites, então vamos exportá-lo ara uma função, a `testaSite`, que recebe o site por parâmetro:

```
// restante do código omitido

func testaSite(site string) {

    resp, _ := http.Get(site)

    if resp.StatusCode == 200 {
        fmt.Println("Site:", site, "foi carregado com sucesso!")
    } else {
        fmt.Println("Site:", site, "está com problemas. Status Code:", resp.StatusCode)
    }
}
```

Agora chamamos essa função dentro do `for`, para cada item do nosso slice. Vamos aproveitar e diminuir a mensagem de teste e adicionar uma linha em branco, para dar um espaçamento entre as mensagens e o menu:

```
// restante do código omitido

func iniciarMonitoramento() {
    fmt.Println("Monitorando...")

    sites := []string{"https://random-status-code.herokuapp.com/",
        "https://www.alura.com.br", "https://www.caelum.com.br"}

    for i, site := range sites {
        fmt.Println("Testando site", i, ":", site)
        testaSite(site)
    }

    fmt.Println("")
}
```

Ao executar o programa, todos os sites do slice são monitorados.

Mas os sites são testados uma única vez e para testarmos novamente, devemos digitar o comando 1. Então, vamos fazer com que os sites sejam testados 5 vezes a cada monitoramento. Para isso, criamos mais um loop:

```
// restante do código omitido

func iniciarMonitoramento() {
    fmt.Println("Monitorando...")

    sites := []string{"https://random-status-code.herokuapp.com/",
```

```
        "https://www.alura.com.br", "https://www.caelum.com.br"}

    for i := 0; i < 5; i++ {
        for i, site := range sites {
            fmt.Println("Testando site", i, ":", site)
            testaSite(site)
        }
    }

    fmt.Println("")
}
```

Aumentando o intervalo entre os monitoramentos

Mas o `for` executa muito rápido, então testaríamos cinco vezes cada site com um intervalo mínimo entre cada teste. Seria interessante termos um intervalo maior entre os testes, por exemplo de 5 em 5 minutos.

Para tal, a cada teste, podemos pedir para o Go esperar um pouco. Fazemos isso utilizando a função `Sleep`, do pacote `time`, passando para ela o quanto de tempo queremos esperar. Representamos o tempo através de constantes da própria biblioteca, como `Second`, `Minute`, entre outras:

```
// restante do código omitido

func iniciarMonitoramento() {
    fmt.Println("Monitorando...")

    sites := []string{"https://random-status-code.herokuapp.com/",
        "https://www.alura.com.br", "https://www.caelum.com.br"}

    for i := 0; i < 5; i++ {
        for i, site := range sites {
            fmt.Println("Testando site", i, ":", site)
            testaSite(site)
        }
        time.Sleep(5 * time.Minute)
    }
    fmt.Println("")
}
```

No caso estamos testando de 5 em 5 minutos, mas esse tempo pode ser aumentado.

Criando constantes

Por último, vamos nos livrar de alguns números do nosso código, e exportá-los para constantes. Por que constantes? Pois elas não podemos ser modificadas.

Os números que queremos atacar é o número 5 que está dentro do `for`, que representa o número de monitoramentos, e o número 5 dentro do `Sleep`, que representa o *delay* do nosso monitoramento.

Então, após os *imports*, criamos as constantes:

```
package main

import (
    "fmt"
    "net/http"
    "os"
    "time"
)

const monitoramentos = 3
const delay = 5

// restante do código omitido
```

Agora, na função **`iniciarMonitoramento`**, utilizamos essas constantes:

```
// restante do código omitido

func iniciarMonitoramento() {
    fmt.Println("Monitorando...")

    sites := []string{"https://random-status-code.herokuapp.com/",
        "https://www.alura.com.br", "https://www.caelum.com.br"}

    for i := 0; i < monitoramentos; i++ {
        for i, site := range sites {
            fmt.Println("Testando site", i, ":", site)
            testaSite(site)
        }

        time.Sleep(delay * time.Second)
    }

    fmt.Println("")
}
```

Agora, se quisermos alterar o *delay* ou a quantidade de monitoramentos, basta alterarmos diretamente na declaração das constantes. E para finalizar, vamos adicionar mais alguns espaçamentos, a cada monitoramento dos nossos sites e depois que o comando foi escolhido:

```
// restante do código omitido

func leComando() int {
    var comandoLido int
    fmt.Scan(&comandoLido)
    fmt.Println("O comando escolhido foi", comandoLido)
    fmt.Println("")

    return comandoLido
}

func iniciarMonitoramento() {
    fmt.Println("Monitorando...")
```

```
        sites := []string{"https://random-status-code.herokuapp.com/",
            "https://www.alura.com.br", "https://www.caelum.com.br"}

        for i := 0; i < monitoramentos; i++ {
            for i, site := range sites {
                fmt.Println("Testando site", i, ":", site)
                testaSite(site)
            }
            time.Sleep(delay * time.Second)
            fmt.Println("")
        }
        fmt.Println("")
    }
```

Com isso, conseguimos monitorar mais de uma vez múltiplos sites.