07

A instrução for

Transcrição

Vamos apagar a função `exibeNomes`, que foi útil para aprendermos o funcionamento do slice, e começar a utilizá-lo na nossa aplicação, para melhorar o armazenamento dos sites a serem monitorados:

```
// 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"}

    site := "https://www.alura.com.br"
    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 que temos uma coleção de sites, podemos monitorar mais de um site ao mesmo tempo. Para isso, devemos percorrer os elementos nosso slice, acessando e monitorando cada site.

Percorrendo os itens do slice

Como vimos na aula anterior, não existe no Go outra estrutura de repetição além do `for`, então vamos utilizá-lo para percorrer os itens do slice de sites. Para cada item do slice, nós vamos mandar uma requisição e testar o seu status code.

Vimos o `for` infinito, que faz com o que o código seja repetido para sempre, mas também podemos utilizar o `for` "tradicional", onde declaramos uma variável e vamos incrementado-a até o tamanho de itens do slice, por exemplo:

```
// 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 < len(sites); i++ {
        fmt.Println(sites[i])
    }

    site := "https://www.alura.com.br"
    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)
    }
}
```

Mas há uma forma mais enxuta de fazer isso nem Go, utilizando o `range`. Ela é como se fosse um operador de iteração do Go, nos dando acesso a cada item do array, ou do slice, e ele nos retorna dos valores, a posição do item iterado e o próprio item daquela posição:

```
// 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("Estou passando na posição", i,
            "do meu slice e essa posição tem o site", site)
    }

    site := "https://www.alura.com.br"
    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 que conseguimos passar por cada item do slice, basta fazer com cada um deles o que já fazemos fora do `for`, ou seja, fazer uma requisição para o site e verificar o seu status code.

Faremos isso no próximo vídeo.