01

Arquivos em Go

Transcrição

O que queremos fazer agora é não ficar dependente do slice, de só monitorar os sites contidos nele. Para isso, teremos um arquivo de texto com todos os sites que queremos monitorar. Adicionaremos vários sites em um arquivo de texto e ao iniciar nosso programa, ele monitorará todos os sites especificados dentro desse arquivo.

Para trabalhar com isso, devemos saber como trabalhamos com arquivo em Go, mais especificamente como lê-los. Para tal, criaremos a função `leSitesDoArquivo`, com essa responsabilidade. Além de ler os dados do arquivo, essa função os retornará em um slice:

```
// restante do código omitido

func leSitesDoArquivo() []string {

}
```

Agora, na função `iniciarMonitoramento`, ao invés de criar o slice na mão, nós criamos a partir do retorna dessa função:

```
// restante do código omitido

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

    // criando o slice a partir da função leSitesDoArquivo()
    sites := leSitesDoArquivo()

    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("")
}
```

Pronto, agora falta implementarmos a função. Antes de ler os dados do arquivo, devemos saber como abri-lo.

Abrindo um arquivo em Go

Para abrir um arquivo em Go, precisamos pedir ao sistema operacional que abra o arquivo para nós. E quem é o representante do sistema operacional em Go? O pacote `os`, que já trabalhamos anteriormente. Nele, há a função `Open`, que abre o arquivo e nos retorna a sua representação em memória e um possível erro que possa ocorrer, que iremos ignorar:

```
// restante do código omitido
```

```
func leSitesDoArquivo() []string {
    arquivo, _ := os.Open("sites.txt")
}
```

Vamos fazer um teste agora, vamos imprimir o arquivo, criar um slice vazio e retorná-lo:

```
// restante do código omitido

func leSitesDoArquivo() []string {

    var sites []string
    arquivo, _ := os.Open("sites.txt")
    fmt.Println(arquivo)

    return sites
}
```

Agora, ao executar o programa e iniciar o monitoramento, vemos uma impressão `<nil>`. O que isso significa? Isso equivale ao `null` de outras linguagens de programação, ou seja, nosso arquivo está nulo. Isso acontece pois estamos abrindo um arquivo que não existe!

Mas como sabemos disso? Sabemos pois foi um erro proposital, já que estamos sendo descuidados, pois estamos ignorando os erros. Em nenhum momentos estamos tratando os erros da linguagem de programação, já que, ao tentar abrir um arquivo que não existe, era para o sistema operacional nos avisar isso, mas estamos ignorando esses avisos, tanto no momento de abrir um arquivo, quanto no momento de realizar as requisições para os sites.

Então, vamos ver como tratar esses erros no próximo vídeo.