Introdução ao Go

Aula 35: Loops Condicionais (Conditional For Loops)

Nesta aula, exploramos uma variação comum do loop `for` em Go, que permite executar o loop enquanto uma condição for verdadeira. Ao invés de definir a condição diretamente dentro dos parênteses do `for`, você pode usar uma expressão booleana (verdadeira ou falsa) como a condição que determinará a continuidade do loop.

Sintaxe do Loop Condicional

A estrutura básica para um loop condicional em Go é:

```go
for someCondition {
    // Execute código enquanto a condição for verdadeira
}
```

- `someCondition`: Pode ser qualquer expressão que resulte em um valor booleano (`true` ou `false`).
- O loop continuará executando o código dentro de seu corpo enquanto a condição for **verdadeira**.
- Quando a condição se tornar **falsa**, o loop será encerrado.

Exemplo de Uso do Loop Condicional

Aqui está um exemplo simples de como usar um loop condicional:

```go
package main

import "fmt"
```

```go
func main() {
    count := 0

    // Loop enquanto count for menor que 5
    for count < 5 {
        fmt.Println("Contagem:", count)
        count++  // Incrementa o contador a cada iteração
    }

    fmt.Println("Loop terminado.")
}
```

Explicação do Exemplo

- O **loop continua enquanto a variável `count` for menor que 5**.
- Em cada iteração, a variável `count` é incrementada, e a condição `count < 5` será avaliada novamente.
- Quando `count` atinge 5, a condição `count < 5` se torna falsa, e o loop é encerrado.

Benefícios do Loop Condicional

- Esse tipo de loop é útil quando não sabemos quantas iterações o loop precisará, mas sabemos que ele deve continuar até que uma condição seja atendida (por exemplo, esperar por uma entrada do usuário ou até que um valor alcance um limite).
- Em comparação com a variação clássica de `for`, onde a condição é explícita, o loop condicional oferece mais flexibilidade ao permitir expressões dinâmicas.

Quando Usar o Loop Condicional

- **Aguardar uma condição ser atendida**: Esse loop pode ser usado quando você precisa aguardar que uma condição externa (como a entrada de dados ou o valor de uma variável) seja atingida.

- **Processamento baseado em eventos**: Por exemplo, um servidor que continua ouvindo requisições até que uma condição de parada seja atendida.

Próximos Passos

🔹 **Prática com Condições Dinâmicas**: Experimente usar esse tipo de loop em cenários reais, como aguardar a resposta de uma API ou processar entradas de usuário. 🔹 **Exploração de outras variações de loops**: Como adicionar controles para garantir que o loop não entre em um ciclo infinito ou otimizar loops com condições complexas.

Se tiver dúvidas ou quiser ver mais exemplos, só avisar!