refactoring.guru

Iterator em Go / Padrões de Projeto

3–4 minutes

O **Iterador** é um padrão de projeto comportamental que permite a passagem sequencial através de uma estrutura de dados complexa sem expor seus detalhes internos.

Graças ao Iterator, os clientes podem examinar elementos de diferentes coleções de maneira semelhante usando uma única interface iterador.

Exemplo conceitual

A ideia principal por trás do padrão Iterator é extrair a lógica de iteração de uma coleção em um objeto diferente denominado iterador. Este iterador fornece um método genérico de iteração sobre uma coleção independente de seu tipo.

collection.go: Coleção

```
package main

type Collection interface {
```

```
    createIterator() Iterator
}
```

userCollection.go: Coleção concreta

```
package main

type UserCollection struct {
    users []*User
}

func (u *UserCollection) createIterator() Iterator {
    return &UserIterator{
        users: u.users,
    }
}
```

iterator.go: Iterador

```
package main

type Iterator interface {
    hasNext() bool
    getNext() *User
}
```

userIterator.go: Iterador concreto

```
package main
```

```
type UserIterator struct {
    index int
    users []*User
}

func (u *UserIterator) hasNext() bool {
    if u.index < len(u.users) {
        return true
    }
    return false

}
func (u *UserIterator) getNext() *User {
    if u.hasNext() {
        user := u.users[u.index]
        u.index++
        return user
    }
    return nil
}
```

user.go: Código cliente

```
package main

type User struct {
    name string
    age  int
}
```

main.go: Código cliente

```
package main

import "fmt"

func main() {

    user1 := &User{
        name: "a",
        age:  30,
    }
    user2 := &User{
        name: "b",
        age:  20,
    }

    userCollection := &UserCollection{
        users: []*User{user1, user2},
    }

    iterator := userCollection.createIterator()

    for iterator.hasNext() {
        user := iterator.getNext()
        fmt.Printf("User is %+v\n", user)
    }
}
```

output.txt: Resultados da execução

User is &{name:a age:30}

User is &{name:b age:20}

Iterator em outras linguagens