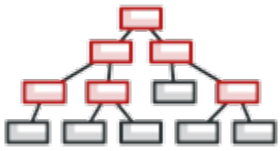
refactoring.guru

Composite em Go / Padrões de Projeto

3–4 minutes



O **Composite** é um padrão de projeto estrutural que permite compor objetos em uma estrutura semelhante a uma árvore e trabalhar com eles como se fosse um objeto singular.

O Composite se tornou uma solução bastante popular para a maioria dos problemas que exigem a construção de uma estrutura em árvore. O grande recurso do Composite é a capacidade de executar métodos recursivamente em toda a estrutura da árvore e resumir os resultados.

Exemplo conceitual

Vamos tentar entender o padrão Composite com um exemplo de sistema de arquivos de um sistema operacional. No sistema de arquivos, existem dois tipos de objetos: arquivos e pastas. Há casos em que arquivos e pastas devem ser tratados da mesma maneira. É aqui que o padrão Composite se torna útil.

Imagine que você precise fazer uma pesquisa por uma

determinada palavra-chave em seu sistema de arquivos. Esta operação de pesquisa se aplica a arquivos e pastas. Para um arquivo, ele apenas examinará o conteúdo do arquivo; para uma pasta, ele percorrerá todos os arquivos dessa pasta para encontrar essa palavra-chave.

component.go: Interface do componente

```
package main

type Component interface {
    search(string)
}
```

folder.go: Composite

```
package main

import "fmt"

type Folder struct {
    components []Component
    name       string
}

func (f *Folder) search(keyword string) {
    fmt.Printf("Serching recursively for keyword %s in folder %s\n",
keyword, f.name)
    for _, composite := range f.components {
        composite.search(keyword)
```

```
    }
}

func (f *Folder) add(c Component) {
    f.components = append(f.components, c)
}
```

file.go: Folha

```
package main

import "fmt"

type File struct {
    name string
}

func (f *File) search(keyword string) {
    fmt.Printf("Searching for keyword %s in file %s\n", keyword,
f.name)
}

func (f *File) getName() string {
    return f.name
}
```

main.go: Código cliente

```
package main
```

```
func main() {
    file1 := &File{name: "File1"}
    file2 := &File{name: "File2"}
    file3 := &File{name: "File3"}

    folder1 := &Folder{
        name: "Folder1",
    }

    folder1.add(file1)

    folder2 := &Folder{
        name: "Folder2",
    }
    folder2.add(file2)
    folder2.add(file3)
    folder2.add(folder1)

    folder2.search("rose")
}
```

output.txt: Resultados da execução

```
Serching recursively for keyword rose in folder Folder2
Searching for keyword rose in file File2
Searching for keyword rose in file File3
Serching recursively for keyword rose in folder Folder1
Searching for keyword rose in file File1
```

Composite em outras linguagens


C#
C++
Java
php
python
Ruby
Rust
TS