refactoring.guru

Command em Go / Padrões de Projeto

3–4 minutes

O **Command** é um padrão de projeto comportamental que converte solicitações ou operações simples em objetos.

A conversão permite a execução adiada ou remota de comandos, armazenamento do histórico de comandos, etc.

Exemplo conceitual

Vejamos o padrão Command com o caso de uma TV. Uma TV pode ser LIGADA por:

- Botão LIGAR no controle remoto;
- Botão LIGAR na própria TV.

Podemos começar implementando o objeto de comando LIGAR com a TV como um receptor. Quando o método `execute` é chamado neste comando, ele, por sua vez, chama a função `TV.on`. A última parte é definir um invocador. Na verdade, teremos dois invocadores: o controle remoto e a própria TV. Ambos irão incorporar o objeto de comando LIGAR.

Observe como empacotamos a mesma solicitação em vários invocadores. Da mesma forma que podemos fazer com outros comandos. A vantagem de criar um objeto de comando separado é que separamos a lógica da IU da lógica do negócio subjacente. Não há necessidade de desenvolver handlers diferentes para cada um dos invocadores. O objeto de comando contém todas as informações de que precisa para ser executado. Portanto, também pode ser usado para uma execução atrasada.

button.go: Invocador

```
package main

type Button struct {
    command Command
}

func (b *Button) press() {
    b.command.execute()
}
```

command.go: Interface do command

```
package main

type Command interface {
    execute()
}
```

onCommand.go: Command concreto

```
package main

type OnCommand struct {
    device Device
}

func (c *OnCommand) execute() {
    c.device.on()
}
```

offCommand.go: Command concreto

```
package main

type OffCommand struct {
    device Device
}

func (c *OffCommand) execute() {
    c.device.off()
}
```

device.go: Interface do receptor

```
package main

type Device interface {
    on()
    off()
}
```

tv.go: Receptor concreto

```
package main

import "fmt"

type Tv struct {
    isRunning bool
}

func (t *Tv) on() {
    t.isRunning = true
    fmt.Println("Turning tv on")
}

func (t *Tv) off() {
    t.isRunning = false
    fmt.Println("Turning tv off")
}
```

main.go: Código cliente

```
package main

func main() {
    tv := &Tv{}

    onCommand := &OnCommand{
        device: tv,
    }
```

```
    offCommand := &OffCommand{
        device: tv,
    }

    onButton := &Button{
        command: onCommand,
    }
    onButton.press()

    offButton := &Button{
        command: offCommand,
    }
    offButton.press()
}
```

output.txt: Resultados da execução

```
Turning tv on
Turning tv off
```

Command em outras linguagens