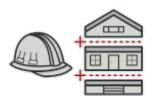
refactoring.guru

Builder em Go / Padrões de Projeto

5-6 minutes



O **Builder** é um padrão de projeto criacional, que permite a construção de objetos complexos passo a passo.

Diferente de outros padrões de criação, o Builder não exige que os produtos tenham uma interface comum. Isso torna possível produzir produtos diferentes usando o mesmo processo de construção.

Exemplo conceitual

O padrão Builder também é usado quando o produto desejado é complexo e requer várias etapas para ser concluído. Nesse caso, vários métodos de construção seriam mais simples do que um único construtor monstruoso. O problema potencial com o processo de construção de vários estágios é que um produto parcialmente construído e instável pode ser exposto ao cliente. O padrão Builder mantém o produto privado até que seja totalmente construído.

No código abaixo, vemos diferentes tipos de casas (igloo e

normalHouse) sendo construídas por iglooBuilder e normalBuilder. Cada tipo de casa tem as mesmas etapas de construção. A struct opcional diretor ajuda a organizar o processo de construção.

iBuilder.go: Interface do builder

```
package main
type IBuilder interface {
  setWindowType()
  setDoorType()
  setNumFloor()
  getHouse() House
}
func getBuilder(builderType string) IBuilder {
  if builderType == "normal" {
     return newNormalBuilder()
  }
  if builderType == "igloo" {
     return newlglooBuilder()
  }
  return nil
}
```

normalBuilder.go: Builder concreto

package main

```
type NormalBuilder struct {
  windowType string
  doorType string
  floor
          int
}
func newNormalBuilder() *NormalBuilder {
  return &NormalBuilder{}
}
func (b *NormalBuilder) setWindowType() {
  b.windowType = "Wooden Window"
}
func (b *NormalBuilder) setDoorType() {
  b.doorType = "Wooden Door"
}
func (b *NormalBuilder) setNumFloor() {
  b.floor = 2
}
func (b *NormalBuilder) getHouse() House {
  return House{
     doorType: b.doorType,
    windowType: b.windowType,
             b.floor,
    floor:
}
```

iglooBuilder.go: Builder concreto

```
package main
type IglooBuilder struct {
  windowType string
  doorType string
  floor
          int
}
func newlglooBuilder() *IglooBuilder {
  return & Igloo Builder{}
}
func (b *IglooBuilder) setWindowType() {
  b.windowType = "Snow Window"
}
func (b *IglooBuilder) setDoorType() {
  b.doorType = "Snow Door"
}
func (b *IglooBuilder) setNumFloor() {
  b.floor = 1
}
func (b *IglooBuilder) getHouse() House {
  return House{
     doorType: b.doorType,
     windowType: b.windowType,
```

```
floor:
              b.floor,
  }
}
house.go: Produto
package main
type House struct {
  windowType string
  doorType string
  floor
           int
}
director.go: Diretor
package main
type Director struct {
  builder IBuilder
}
func newDirector(b IBuilder) *Director {
  return & Director{
     builder: b,
  }
}
func (d *Director) setBuilder(b IBuilder) {
  d.builder = b
```

```
}
func (d *Director) buildHouse() House {
  d.builder.setDoorType()
  d.builder.setWindowType()
  d.builder.setNumFloor()
  return d.builder.getHouse()
}
main.go: Código cliente
package main
import "fmt"
func main() {
  normalBuilder := getBuilder("normal")
  iglooBuilder := getBuilder("igloo")
  director := newDirector(normalBuilder)
  normalHouse := director.buildHouse()
  fmt.Printf("Normal House Door Type: %s\n",
normalHouse.doorType)
  fmt.Printf("Normal House Window Type: %s\n",
normalHouse.windowType)
  fmt.Printf("Normal House Num Floor: %d\n", normalHouse.floor)
  director.setBuilder(iglooBuilder)
  iglooHouse := director.buildHouse()
```

```
fmt.Printf("\nlgloo House Door Type: %s\n",
iglooHouse.doorType)
  fmt.Printf("Igloo House Window Type: %s\n",
iglooHouse.windowType)
  fmt.Printf("Igloo House Num Floor: %d\n", iglooHouse.floor)
}
```

output.txt: Resultados da execução

Normal House Door Type: Wooden Door

Normal House Window Type: Wooden Window

Normal House Num Floor: 2

Igloo House Door Type: Snow Door

Igloo House Window Type: Snow Window

Igloo House Num Floor: 1

Builder em outras linguagens



















6/24/24, 12:36 7 of 7