



[🏠](#) / [Паттерны проектирования](#) / [Строитель](#) / [Go](#)



Строитель на Go

Строитель — это порождающий паттерн проектирования, который позволяет создавать объекты пошагово.

В отличие от других порождающих паттернов, Строитель позволяет производить различные продукты, используя один и тот же процесс строительства.

 [Подробнее о паттерне Строитель →](#)

Навигация

 [Интро](#)

 [Концептуальный пример](#)

 [iBuilder](#)

 [normalBuilder](#)


 [iglooBuilder](#)

[Главная](#) [Рефакторинг](#) [Паттерны](#) [Премиум контент](#)
[Форум](#) [Связаться](#)



© 2014-2023 Refactoring.Guru.

Все права защищены.

 Иллюстрации нарисовал Дмитрий Жарт

[Условия использования](#)

[Политика конфиденциальности](#)

[Использование контента](#) [About us](#)

Паттерн Строитель также используется, когда нужный продукт сложный и требует нескольких шагов для построения. В таких случаях несколько конструкторных методов подойдут лучше, чем один громадный конструктор. При использовании пошагового построения объектов потенциальной проблемой является выдача клиенту частично построенного нестабильного продукта. Паттерн "Строитель" скрывает объект до тех пор, пока он не построен до конца.

Нижеприведенный код использует разные типы домов (`igloo` и `normalHouse`), которые конструируются с помощью строителей `iglooBuilder` и `normalBuilder` . При создании каждого дома используются одинаковые шаги. Для помощи в организации процесса можно использовать директор, хоть это и необязательно.

iBuilder.go: Интерфейс строителя

```
package main

type IBuilder interface {
    setWindowType()
    setDoorType()
    setNumFloor()
    getHouse() House
}

func getBuilder(builderType string) IBuilder {
    if builderType == "normal" {
        return newNormalBuilder()
    }

    if builderType == "igloo" {
        return newIglooBuilder()
    }
    return nil
}
```

normalBuilder.go: Конкретный строитель

```
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,
            floor:     b.floor,
        }
    }
}

```

iglooBuilder.go: Конкретный строитель

```

package main

type IglooBuilder struct {
    windowType string
    doorType   string
    floor      int
}

func newIglooBuilder() *IglooBuilder {
    return &IglooBuilder{}
}

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: Продукт

```

package main

type House struct {
    windowType string
    doorType   string
    floor      int
}

```

director.go: Директор

```

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: Клиентский код

```

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("\nIgloo 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: Результат выполнения

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

По материалам: *Golang By Example*

ВЕРНУТЬСЯ НАЗАД

ЧИТАЕМ ДАЛЬШЕ

← Абстрактная фабрика на Go

Фабричный метод на Go →

Строитель на других языках программирования

