# 访问者模式

## 一、基本概念 **1、定义：**

##### 访问者模式（Visitor），表示了一个作用域某对象结构中的各元素的操作，它可以使你在不改变各元素的类的前提下定义作用于这些元素的新操作。

##### 优势是把数据结构和作用于数据结构之间的操作进行解耦合操作，使得操作可以比较自由的进行演化。

### **劣势是该方法比较复杂。**

### 2、组成环节：

##### 1、Visitor(访问者类)：为该对象结构中ConcreteElement的每一个类声明一个visit操作。

##### 2、ConcreteVisitor(具体访问者类)：实现每一个visitor声明的具体操作。

##### 3、ObjectStructure(枚举类)：用于枚举其中的元素，可以提供一个接口哎访问其中的所有元素。

##### 4、Element(元素类)：定义一个accept的操作。以某个访问者作为参数。

##### 5、ConcreteElement(具体元素类)：实现具体的accept方法。

## 二、go语言中实现

##### 1、首先定义一个Element类型，在对其子类当中的方法进行定义。

type Element interface {  
 Accept(v visitor)  
}  
  
type ConcreteElementA struct{}  
  
type ConcreteElementB struct{}  
  
func (ca \*ConcreteElementA) Accept(v visitor) {  
 v.VisitConcreteElementA()  
}  
  
func (cb \*ConcreteElementB) Accept(v visitor) {  
 v.VisitConcreteElementB()  
}

##### 2、再定义一个visitor类，并且对其子类的实现方法进行实现。

type visitor interface {  
 VisitConcreteElementA()  
 VisitConcreteElementB()  
}  
  
type ConcreteVisitorA struct{}  
  
type ConcreteVisitorB struct{}  
  
func (ca ConcreteVisitorA) VisitConcreteElementA() {  
 println("visitorA on ElementA.")  
}  
  
func (ca ConcreteVisitorA) VisitConcreteElementB() {  
 println("visitorA on ElementB.")  
}  
  
func (cb ConcreteVisitorB) VisitConcreteElementA() {  
 println("visitorB on ElementA.")  
}  
  
func (cb ConcreteVisitorB) VisitConcreteElementB() {  
 println("visitorB on ElementB.")  
}

##### 3、定义ObjectStructure类，并且声明其中的实现方法。

type ObjectStructure struct{}  
  
func (o ObjectStructure) Display(v visitor) {  
 ca := ConcreteElementA{}  
 cb := ConcreteElementB{}  
  
 ca.Accept(v)  
 cb.Accept(v)  
}

##### 4、开始进行main方法的调用。

func main() {  
 o := ObjectStructure{}  
 va := ConcreteVisitorA{}  
 vb := ConcreteVisitorB{}  
 o.Display(va)  
 o.Display(vb)  
}

##### 5、结果的输出。

visitorA on ElementA.

visitorA on ElementB.

visitorB on ElementA.

visitorB on ElementB.