# 职责链模式

## 一、基本概念 **1、定义：**

##### 职责链模式（Responsibility），使多个对象都有机会处理请求，从而避免请求的发送者和接收者之间的耦合关系。将这个对象连城一条链，并沿着这条链传递该请求，直到有一个对象处理它为止。

##### 优势是当用户提出一个请求的时候，可以沿着一条链传递到一个具体的对象来处理他。仅需要保持一个指向其后继者的引用，而不需要保持它所有的候选接收者的引用，增强了灵活性。

### 2、组成环节：

##### Handle(处理者类):定义一个处理请示的接口。

##### ConcreteCommand(具体处理者类):处理自己负责的请求，可以访问它的后继者，如果可以处理该请求，就进行处理，否则转发给其后继者。

## 二、go语言中实现

##### 1、首先定义Handle类，并列出该类型所拥有的方法。

type Handle interface {  
 SetSeccessor(h Handle)  
 HandleRequest(request int)  
}

##### 2、对ConcreteCommand方法进行定义，并且对中各类的实现方法进行讲述。

type ConcreteHandleA struct {  
 Handle  
}  
  
type ConcreteHandleB struct {  
 Handle  
}  
  
type ConcreteHandleC struct {  
 Handle  
}  
  
func (ca \*ConcreteHandleA) SetSeccessor(h Handle) {  
 ca.Handle = h  
}  
  
func (cb \*ConcreteHandleB) SetSeccessor(h Handle) {  
 cb.Handle = h  
}  
  
func (cc \*ConcreteHandleC) SetSeccessor(h Handle) {  
 cc.Handle = h  
}  
  
func (ca \*ConcreteHandleA) HandleRequest(request int) {  
 if request < 10 {  
 println("use HandleA.")  
 } else {  
 ca.Handle.HandleRequest(request)  
 }  
}  
  
func (cb \*ConcreteHandleB) HandleRequest(request int) {  
 if request >= 10 && request < 20 {  
 println("use HandleB.")  
 } else {  
 cb.Handle.HandleRequest(request)  
 }  
}  
  
func (cc \*ConcreteHandleC) HandleRequest(request int) {  
 if request > 20 {  
 println("use HandleC.")  
 } else {  
 cc.Handle.HandleRequest(request)  
 }  
}

##### 通过main方法进行client端调用。

func main() {  
 ca := ConcreteHandleA{}  
 cb := ConcreteHandleB{}  
 cc := ConcreteHandleC{}  
  
 ca.SetSeccessor(&cb)  
 cb.SetSeccessor(&cc)  
  
 list := []int{1, 11, 21}  
  
 for \_, num := range list {  
 ca.HandleRequest(num)  
 }  
}

##### 输出结果

**use HandleA.**

**use HandleB.**

**use HandleC.**