# 状态模式

## 一、基本概念 **1、定义：**

##### 状态模式（State Pattern）允许一个环境类根据实际情况修改，类的行为基于它的状态改变。

##### 优势是可以随时对context方法里面的类进行修改，灵活性较高，而且具体状态类和context解耦合，互不影响

### 2、组成环节：

##### 1、State (抽象状态):具体状态的父类，为context获取具体状态来实现一个接口

##### 2、ConcreteState(具体状态)：抽象状态的子类，为具体的状态，在具体调用的时候使用

##### 3、Context(环境)：某个带有状态的类，可以通过调用state类去改变自己的状态

## 二、go语言中实现

##### 1、首先定义一个State类型

type State interface {  
 Do()  
}

##### 2、在定义两个具体类型的state类

type StateA struct {}  
  
type StateB struct {}

##### 3、定义一个调用的context的类

type Context struct {  
 State  
}

##### 4、对context方法的具体实现

func (c \*Context)Set(s State) {  
 c.State = s  
}  
  
func (c \*Context)Get() State {  
 return c.State  
}

##### 5、对具体的state进行do的实现。

func (s StateA)Do() {  
 fmt.Println("stateA is doing.")  
}  
  
func (s StateB)Do() {  
 fmt.Println("stateB is doing.")  
}

##### 6、开始进行方法的调用。

func main() {  
 var c Context  
 var sa StateA  
 c.Set(sa)  
 c.Get().Do()  
 var sb StateB  
 c.Set(sb)  
 c.Get().Do()  
}

##### 7、结果的输出。

stateA is doing.

stateB is doing.