
观察者模式

一、什么是观察者模式？

观察者模式即 Observer Pattern。观察者模式的主旨是定义对象间的一种一（被观察者 Observable）对多（观察者 Observer）的依赖关系，当一个对象的状态发生改变时，所有依赖于它的对象都得到通知并被自动更新。因此，观察者模式中主要有观察者（Observer）和被观察者（Observable）2 种对象。其中，被观察者（Observable）是个抽象类或接口，只能被继承，观察者（Observer）并不是一个类而是一个接口，所以观察者可以有多个，实现了该接口的类都属于观察者。观察者也可以同时是被观察者，被观察者同样可以是观察者。

二、观察者模式用来解决哪些问题？

举个例子，在 Android 中，我们希望数据的变化自动更新所有相关 UI，那么这里的数据就是一个被观察的对象，所有涉及相关数据的 UI 就是多个观察者，他们之间就是一对多的依赖关系，并且我们希望做到“一”变化时，依赖这个“一”的“多”也能够同步改变。MVC 就是观察者模式的一个实例，在 Android 中界面控件的事件响应以及 BroadcastReceiver 等都是基于观察者模式来实现的。

三、如何使用观察者模式？

在一个被观察者中需要提供注册和删除观察者对象的接口，观察者需要为那些在目标发生改变时，获得通知的对象定义更新的接口。

四、观察者模式使用示例及步骤：

下面是我们就利用观察者实现 Android 中 Button 按钮的监听处理，Button 对象就是我们要观察的目标对象，setOnClickListener()方法就是注册一个观察者，具体代码如下：

步骤 1：定义抽象的观察者

```
public interface Listener {  
  
    //起到和更新一样的用途  
  
    void onClick();  
  
}
```

步骤 2：定义抽象的被观察者

```
public interface Listened {  
  
    //添加观察者  
  
    void setOnClickListener(Listener w);  
  
    //移除观察者  
  
    void removeClickListener(Listener w);  
  
    //通知观察者  
  
    void notifyClickListeners();  
  
}
```

步骤 3：定义具体的观察者继承抽象的观察者

```
public class OnClickListener implements Listener {  
  
    public void onClick() {  
  
        System.out.println("又被点击了 !!! ");  
  
    }  
  
}
```

步骤 4：定义具体的被观察者继承自抽象的被观察者

```
public class Button implements Listened {  
  
    private List<Listener> list = new ArrayList<Listener>();  
  
    public void setOnClickListener(Listener l) {  
        list.add(l);  
    }  
  
    public void removeClickListener(Listener l) {  
        list.remove(l);  
    }  
  
    public void notifyClickListeners() {  
        for (Listener l : list) {  
            l.onClick();  
        }  
    }  
}
```

步骤 5：测试类

```
public class Test {  
  
    public static void main(String[] args) {  
  
        //创建被观察者对象  
  
        Button btn = new Button();
```

```
//创建观察者对象
```

```
OnClickListener clicklistener = new OnClickListener();
```

```
OnClickListener clicklistener2 = new OnClickListener();
```

```
//添加到观察者集合中
```

```
btn.setOnClickListener(clicklistener);
```

```
btn.setOnClickListener(clicklistener2);
```

```
//调用通知方法
```

```
btn.notifyClickListeners();
```

```
}
```

```
}
```