853769620

observer模式

场景

（1）比如我们有一个目标对象，一旦这个目标对象的状态改变了，然后的话就去通知相关的对象，我的状态改变了，数据也变化了

 【观察目标】

public class **Subject extends Observable** {

   private String message;

   public String getMessage() {

       return message;

  }

   public void changed() {

       this.setChanged();

       this.notifyObservers(message); // 推模式

       // this.notifyObservers(); // 拉模式

  }

}

【观察者】

public class **ConcreteObserver implements Observer** {

   public void update(Observerable observable, Object object) {

       System.out.println(object); // 获取推模式推送过来的数据

       // System.out.println(((Subject)observable).getMessage()); // 拉模式情况下自己去拿数据

  }

}

前端开发，js脚本，要对界面上的各种视图组件，表格点击事件，按钮点击，添加监听器，一旦某种事件发生，就会执行监听器中的方法

观察者模式还是**非常常用**的，不过说实话，常见于基于zookeeper进行分布式系统之间的协调工作，比如分布式锁的注册以及监听是否释放。还有就是两个系统之间如果做了异步的处理，那么如果A系统发送异步请求给了B系统，但是要得到B系统的一个状态改变的消息，可以采用观察者模式。

基于zookeeper去做分布式锁

（1）系统A尝试获取zookeeper上的一个锁，获取到了

（2）系统B尝试获取zookeeper上的一个锁，被系统A给锁了，没有获取到锁，此时系统B在zookeeper上可以**注册一个监听器（观察者）**

（3）系统A一旦将锁给释放了，zookeeper感受到锁被释放了，就会立即通知系统B注册的那个监听器

（4）系统B就立即被通知到了，系统A释放了锁，系统B可以重新尝试在zookeeper上加锁

我们的电商系统里，也是有这种场景的，如果两个系统之间走了异步请求，那么可以基于上面那种观察者模式现在一个进程内实现监听，以后拆分微服务分布式架构了，可以改成基于zookeeper来做分布式协调。

系统A发送了一条消息到内存队列，系统B获取了消息开始执行操作

但是系统A需要知道系统B的一个执行的结果如何，此时怎么办？

系统A需要注册一个观察者到系统B上去，系统B执行完了之后，将执行的结果，反过来通知给系统

我们就可以基于观察者模式去做