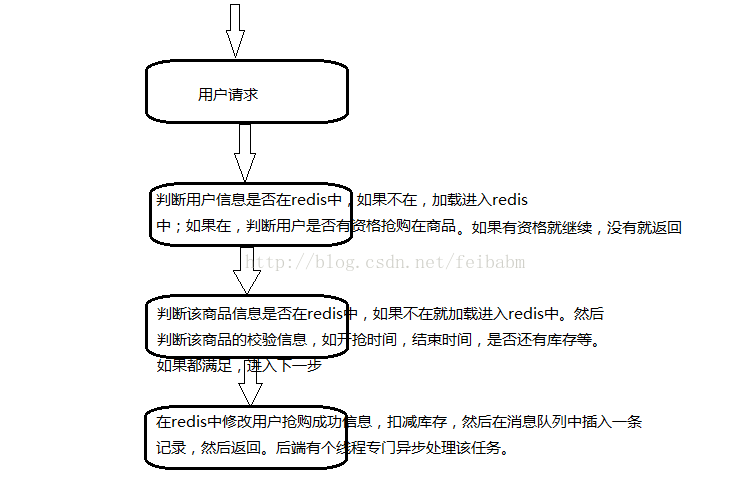
传入：用户电话即可。



实现功能：

查询是否有预约：

Redis查询，若有则可以秒杀，若没有，则去数据库查询，若有预约，则把信息加入Redis。若没有，则返还没有预约信息。

查询商品是否已经加入redis：

如果没有，则直接加入，如时间，库存等。

在redis查询是否还有库存：

有则抢购成功，扣库存，用redis充当**消息队列插入用户秒杀记录，秒杀结束再同步处理（后台线程异步处理或者开线程池处理）**。没有库存则返回错误信息。

代码实现：

1. **public** SecKillResult secKillProduct(String userPhone, **long** productId) {
2. String state = (String)redisTemplate.opsForValue().get(userPhone + "\_"+ productId);
3. //用户信息加载
4. **if**(**null** == state){
5. SuccessKilled successKilled = **new** SuccessKilled();
6. successKilled.setSeckillId(productId);
7. successKilled.setUserPhone(Long.valueOf(userPhone));
8. successKilled = successKilledMapper.selectOne(successKilled);
9. **if**(**null** == successKilled){
10. **return** **new** SecKillResult(**false**, "该用户没有预约");
11. }**else**{
12. **synchronized** (**this**){
13. state = (String)redisTemplate.opsForValue().get(userPhone + "\_"+ productId);
14. **if**(**null** == state){
15. redisTemplate.opsForValue().set(userPhone + "\_" + productId, successKilled.getState().toString(), 300, TimeUnit.SECONDS);
16. state = String.valueOf(successKilled.getState());
17. }
18. }
20. }
21. }
22. **if**(state.equals("-1")){
23. //查询产品信息
24. //            ProductInfo productInfo = (ProductInfo)redisTemplate.opsForValue().get(productId + "");
25. List values = redisTemplate.opsForHash().values(productId + "");
26. **if**(values.size() == 0){
27. Seckill seckill = seckillMapper.selectByPrimaryKey(productId);
28. **if**(**null** == seckill){
29. **return** **new** SecKillResult(**false**, "没有该秒杀商品信息");
30. }
31. **synchronized** (**this**){
32. **if**(!redisTemplate.opsForHash().hasKey(productId + "", "number")){
33. //                        productInfo = new ProductInfo(seckill.getSeckillId(), seckill.getNumber(), seckill.getStartTime(), seckill.getEndTime());
34. HashMap<String, String> productHash = **new** HashMap<>();
35. productHash.put("number", seckill.getNumber() + "");
36. productHash.put("startTime", seckill.getStartTime().getTime() + "");
37. productHash.put("endTime", seckill.getEndTime().getTime() + "");
38. redisTemplate.opsForHash().putAll(productId +"", productHash);
39. redisTemplate.expire(productId + "", 300, TimeUnit.SECONDS);
40. values = redisTemplate.opsForHash().values(productId + "");
41. }
42. }
43. }
44. **if**( **new** Date(Long.valueOf((String)values.get(1))).after(**new** Date(System.currentTimeMillis()))){
45. **return** **new** SecKillResult(**false**, "抢购还没有开始");
46. } **else** **if**(**new** Date(Long.valueOf((String)values.get(2))).before(**new** Date(System.currentTimeMillis()))){
47. **return** **new** SecKillResult(**false**, "抢购已经结束");
48. } **else** {
49. Long userState = redisTemplate.opsForValue().increment(userPhone + "\_" + productId, 1);
50. **if**(userState == 0){
51. //                    Long increment = redisTemplate.opsForValue().increment(productId, -1);
52. Long number = redisTemplate.opsForHash().increment(productId + "", "number", -1);
53. **if**(number >= 0){
54. //消息队列异步更新库存，以及用户的预约信息
55. QueueEntity queueEntity = **new** QueueEntity(userPhone, productId);
56. ExecutorPool.queue.offer(queueEntity);
57. }**else** {
58. **return** **new** SecKillResult(**false**, "商品已经抢购完成");
59. }
60. }**else** {
61. redisTemplate.opsForValue().increment(userPhone + "\_" + productId, -1);
62. **return** **new** SecKillResult(**false**, "您已抢购过该产品");
63. }
64. }
65. } **else** {
66. **return** **new** SecKillResult(**false**, "您已抢购过该产品");
67. }
68. **return** **null**;
69. }