|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | OrderService.AddOrder | OrderDao.insertOrder | OrderItemDao.AddOrderItem | result |
| 1 | 正常 | 正常 | 正常 | 正常 |
| 2 | result = 10 / 0 | 正常 | 正常 | 整个事务回滚 |
| 3 | 正常 | a = 10 / 0 | 正常 | 整个事务回滚 |
| 4 | 正常 | 正常 | a = 10 / 0 | 整个事务回滚 |
| 5 | 正常 | 正常 | 正常 ***REQUIRES\_NEW*** | 正常 |
| 6 | result = 10 / 0，在OrderItemDao.AddOrderItem之前 | 正常 | 正常 ***REQUIRES\_NEW*** | 整个事务回滚 |
| 7 | result = 10 / 0，在OrderItemDao.AddOrderItem之后 | 正常 | 正常 ***REQUIRES\_NEW*** | OrderItemDao.AddOrderItem成功，OrderDao.insertOrder和OrderService.AddOrder事务回滚 |
| 8 | 正常 | a = 10 / 0 | 正常 ***REQUIRES\_NEW*** | 整个事务回滚 |
| 9 | 正常 | 正常 | a = 10 / 0  ***REQUIRES\_NEW*** | OrderDao.insertOrder和OrderService.AddOrder成功，OrderItemDao.AddOrderItem事务回滚 |

1.

原因：

不使用REQUIRES\_NEW时，三个操作属于一个事务，出现报错时会作为一个整体回滚，使用REQUIRES\_NEW时，OrderItemDao.AddOrderItem单独属于一个事务，两个事务分别报错，只会回滚对应的事务。

本项目中应当全部使用REQUIRED属性，将三个操作置于同一事务，保证三个操作作为一个整体。

2.

1. 消息监听

优点：

简单易实现：不需要额外的连接管理，基本的HTTP请求就能实现。

兼容性好：几乎所有浏览器和环境都支持HTTP请求，且易于调试和维护。

缺点：

资源消耗大：频繁发送请求会造成服务器和网络带宽负担。

不适合高并发场景：频繁轮询在高并发场景下会加重服务器负担。

2. AJAX 轮询

优点：

实现较为简单：AJAX轮询技术成熟。

兼容性强：所有现代浏览器都支持AJAX请求。

请求频率可控：可以通过调整轮询间隔，来权衡实时性和服务器压力。

缺点：

延迟性：无法做到实时推送，存在一定的延迟，特别是在轮询频率较低的情况下。

资源浪费：即使没有状态更新，前端也会继续发送请求，浪费服务器和带宽资源。

服务器负载较高：在高并发情况下，频繁的请求会加重服务器压力。

3. WebSocket

优点：

实时性强：消息一旦产生会立即推送到前端，真正的实时更新。

高效：与轮询相比，WebSocket只在有新数据时传输，因此通信开销较小，节省了带宽和服务器资源。

缺点：

连接管理复杂：WebSocket需要保持长连接，前端和后端都需要处理连接断开、重连等情况，增加了实现难度。

浏览器兼容性问题：在一些老旧设备或浏览器中，可能存在不兼容问题。

在本项目中，采用WebSocket实现。前端发送下订单请求后，会通过SockJS连接WebSocket，并监听一个包含用户userId的域名，后端成功下订单后，会使用convertAndSendToUser函数，将处理的结果发到对应用户的端口。