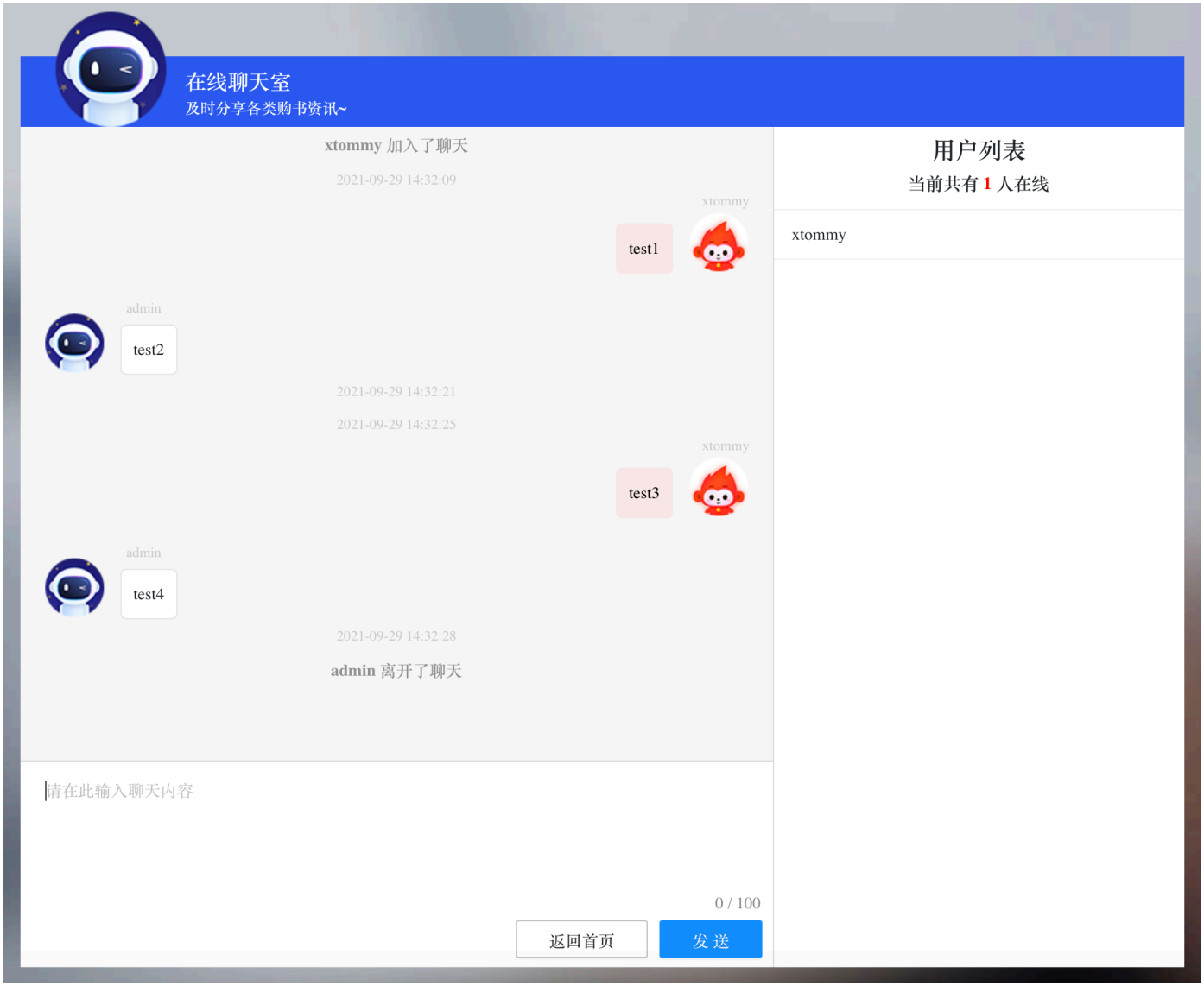
在线聊天室

测试效果如图所示，功能均正常。用户登录后，才能进行聊天。由于webSocket与http session并不互通，因此配置了相关的代码，在webSocket中获取当前的http session。



观察不同事务属性的差异

为了测试不同事务属性下是否触发了回滚，我们修改了原有的下订单逻辑，拆分为四个步骤。

原先的逻辑为**先判断订单是否合法，再下单。**系统检测库存足够后，才会添加订单，并减少书的库存。现在修改为**系统先添加订单，再减少库存，库存不足时应该触发回滚。**

下订单的逻辑如下所示：

1. 添加订单：cartDao.addOrder
2. 添加订单项：cartDao.addOrderItem
3. 删除购物车：cartDao.deleteCart
4. 更新图书库存：cartDao.updateBookInventory

由于不需要测试事务隔离级别带来的差异，因此我们统一使用默认的READ_COMMITTED。

1.REQUIRED

我们对service层的submitCart方法使用**REQUIRED属性**，并对dao层的四个方法同样使用**REQUIRED属性**。

程序进入service层的submitCart方法后，会开启一个新的事务。由于dao层的方法中都采用了REQUIRED属性，因此会在**同一个事务中执行dao层的方法**。若dao层的四个方法均没有出错，则提交事务，完成下订单的操作。若dao层中有方法在执行过程中出现了错误，则整个事务会回滚，保证操作的一致性和完整性。

例如，我们购买11本同样的书，该书的库存量为10的书，则会出现以下报错。

```
2021-09-29 14:33:33.396  INFO 65288 --- [enerContainer-1] com.bookstore.daoimpl.CartDaoImpl        : addOrderItem dao completed.
2021-09-29 14:33:33.397  INFO 65288 --- [enerContainer-1] com.bookstore.daoimpl.CartDaoImpl        : deleteCart dao executed.
Hibernate: delete from `cart_item` where `user_id`=?
2021-09-29 14:33:33.399  INFO 65288 --- [enerContainer-1] com.bookstore.daoimpl.CartDaoImpl        : deleteCart dao completed.
2021-09-29 14:33:33.399  INFO 65288 --- [enerContainer-1] com.bookstore.daoimpl.CartDaoImpl        : updateBookInventory dao executed.
Hibernate: select book0_.book_id as book_id1_0_, book0_.author as author2_0_, book0_.brief as brief3_0_, book0_.description as descript4_0_, book0_.enabled as enabled5_0_, book0_.image as image6_0_,
Hibernate: update `book` set `inventory`=? where `book_id`=?
2021-09-29 14:33:33.420  WARN 65288 --- [enerContainer-1] o.h.engine.jdbc.spi.SqlExceptionHelper   : SQL Error: 1264, SQLState: 22001
2021-09-29 14:33:33.420 ERROR 65288 --- [enerContainer-1] o.h.engine.jdbc.spi.SqlExceptionHelper   : Data truncation: Out of range value for column 'inventory' at row 1
2021-09-29 14:33:33.438 ERROR 65288 --- [enerContainer-1] com.bookstore.listener.OrderListener     : could not execute statement; SQL [n/a]; nested exception is org.hibernate.exception.DataExceptio
```

可以看到，由于计算出的更新后的库存量为负数，而sql中的inventory字段为**unsigned int**，因此执行sql时会抛出异常，**整个事务会回滚，数据库回到下订单前的状态**。

2.NOT_SUPPORTED

对于更新库存的方法，即updateBookInventory，我们使用**NOT_SUPPORTED属性**。其余方法我们仍然使用**REQUIRED属性**。

采用NOT_SUPPORTED属性，容器不会为这个方法开启事务。在执行该方法前，会判断当前线程是否存在事务，如果存在则挂起当前事物，并执行查询时使用新的连接。

我们依然购买11本同样的书，该书的库存量为10。

```
2021-09-29 16:15:17.599  INFO 78267 --- [enerContainer-1] com.bookstore.daoimpl.CartDaoImpl        : addOrderItem dao completed.
2021-09-29 16:15:17.600  INFO 78267 --- [enerContainer-1] com.bookstore.daoimpl.CartDaoImpl        : deleteCart dao executed.
Hibernate: delete from `cart_item` where `user_id`=?
2021-09-29 16:15:17.605  INFO 78267 --- [enerContainer-1] com.bookstore.daoimpl.CartDaoImpl        : deleteCart dao completed.
2021-09-29 16:15:17.607  INFO 78267 --- [enerContainer-1] com.bookstore.daoimpl.CartDaoImpl        : updateBookInventory dao executed.
Hibernate: select book0_.book_id as book_id1_0_, book0_.author as author2_0_, book0_.brief as brief3_0_, book0_.description as descript4_0_, book0_.enabled as enabled5_0_, book0_.image as image6_0_, bo
Hibernate: update `book` set `inventory`=? where `book_id`=?
preHandle Success
Hibernate: select user0_.user_id as user_id1_5_, user0_.email as email2_5_, user0_.enabled as enabled3_5_, user0_.name as name4_5_ from user user0_ where user0_.user_id=?
2021-09-29 16:16:08.756  WARN 78267 --- [enerContainer-1] o.h.engine.jdbc.spi.SqlExceptionHelper   : SQL Error: 1205, SQLState: 40001
2021-09-29 16:16:08.756 ERROR 78267 --- [enerContainer-1] o.h.engine.jdbc.spi.SqlExceptionHelper   : Lock wait timeout exceeded; try restarting transaction
2021-09-29 16:16:08.770 ERROR 78267 --- [enerContainer-1] com.bookstore.listener.OrderListener     : could not execute statement; SQL [n/a]; nested exception is org.hibernate.PessimisticLockException:
```

可以看到，在执行NOT_SUPPORTED方法后，容器会将当前事务挂起。由于NOT_SUPPORTED方法中sql语句无法正常执行，因此lock wait会超时，抛出异常，**并会导致其他事务回滚**。因此这个例子中结果与使用REQUIRED一致，数据库同样回到下订单前的状态。

3.MANDATORY

MANDATORY属性要求**当前的方法必须运行在事物内部**。如果没有正在运行的事务，则抛出异常。

我们对service层的submitCart方法使用**MANDATORY属性**，其余方法保持为**REQUIRED属性**。

```
com.bookstore.controller.CartController@64d2967
com.bookstore.serviceimpl.CartServiceImpl@cc06797
received order at time:2021-09-29 16:34:42 userId:1
2021-09-29 16:34:42.986 ERROR 82239 --- [enerContainer-1] com.bookstore.listener.OrderListener     : No existing transaction found for transaction marked with propagation 'mandatory'
```

可以看到，程序抛出了异常。这是因为调用submitCart方法时没有正在运行的事务，因此抛出异常。

4.REQUIRED_NEW

REQUIRED_NEW属性要求当前的方法必须启动新事务，并在它自己的事务内运行。如果有事务正在运行，应该将它挂起。

我们对dao层的添加订单方法，即addOrder，使用**REQUIRED_NEW属性**，其余方法保持为**REQUIRED属性**。因此dao层中其他三个方法属于同一事务，而添加订单方法则会属于新的事务。

```
2021-09-29 16:42:31.726  INFO 83620 --- [enerContainer-1] com.bookstore.daoimpl.CartDaoImpl        : addOrderItem dao completed.
2021-09-29 16:42:31.726  INFO 83620 --- [enerContainer-1] com.bookstore.daoimpl.CartDaoImpl        : deleteCart dao executed.
Hibernate: delete from `cart_item` where `user_id`=?
2021-09-29 16:42:31.729  INFO 83620 --- [enerContainer-1] com.bookstore.daoimpl.CartDaoImpl        : deleteCart dao completed.
2021-09-29 16:42:31.729  INFO 83620 --- [enerContainer-1] com.bookstore.daoimpl.CartDaoImpl        : updateBookInventory dao executed.
Hibernate: select book0_.book_id as book_id1_0_, book0_.author as author2_0_, book0_.brief as brief3_0_, book0_.description as descript4_0_, book0_.enabled as enabled5_0_, book0_.image as image6_0_, b
Hibernate: update `book` set `inventory`=? where `book_id`=?
2021-09-29 16:42:31.749  WARN 83620 --- [enerContainer-1] o.h.engine.jdbc.spi.SqlExceptionHelper   : SQL Error: 1264, SQLState: 22001
2021-09-29 16:42:31.750 ERROR 83620 --- [enerContainer-1] o.h.engine.jdbc.spi.SqlExceptionHelper   : Data truncation: Out of range value for column 'inventory' at row 1
2021-09-29 16:42:31.767 ERROR 83620 --- [enerContainer-1] com.bookstore.listener.OrderListener     : could not execute statement; SQL [n/a]; nested exception is org.hibernate.exception.DataException: 
```

可以看到，程序同样抛出了异常。由于添加订单项、删除购物车和更新图书库存属于同一事务，因此**出错时三者一起回滚**。然而添加订单属于新的事务，**因此不会受到影响，事务正常提交**。

	order_id 1	user_id	time	price
1	13	1	2021-09-29 16:42:32	315.00