

hw1: Stateful Service & Messaging

task1

有状态

无状态

结论

Task2

结果

task1

1. 优化服务层设计，确定有状态与无状态服务的运用方式，要求至少分别选择一个服务设计成有状态的和无状态的，并观察它们的差异，确认你的设计合理。

有状态

选择CartServiceImpl 进行设计，有状态情况下，设置为@Scope("session")

```
@Service
@Scope("session")
public class CartServiceImpl implements CartService {
    @Autowired
    private CartDao cartDao;

    @Override
    public List<Cart> getCart() { return cartDao.getCart(); }

    @Override
    public List<Cart> clearCart() { return cartDao.clearCart(); }

    @Override
    public Cart addToCart(String name, String author, Integer price, Integer number, Integer bookId, Integer userId) {
        return cartDao.addToCart(name, author, price, number, bookId, userId);
    }

    @Override
    public List<Cart> getUserCart(Integer userId) { return cartDao.getUserCart(userId); }
}
```

```

@RestController
@Scope("session")
public class CartController {

    @Autowired
    private CartService cartService;

    @Autowired
    WebApplicationContext applicationContext;

    @GetMapping("/addToCart")
    public Cart addToCart(
        @RequestParam(required = false) String name,
        @RequestParam(required = false) String author,
        @RequestParam(required = false) Integer price,
        @RequestParam(required = false) Integer number,
        @RequestParam(required = false) Integer bookId,
        @RequestParam(required = false) Integer userId
    ){
        CartService cartService = applicationContext.getBean(CartService.class);
        System.out.println(cartService);
        System.out.println(this);
        return cartService.addToCart(name, author, price, number, bookId, userId);
    }
}

```

测试结果如下

第一次使用Google chrome浏览器

com.reins.bookstore.serviceimpl.CartServiceImpl@2824e8bb

com.reins.bookstore.controller.CartController@4c5c9e37

第二次使用Google chrome浏览器

com.reins.bookstore.serviceimpl.CartServiceImpl@2824e8bb

com.reins.bookstore.controller.CartController@4c5c9e37

第一次使用Safari浏览器

com.reins.bookstore.serviceimpl.CartServiceImpl@2f097180

com.reins.bookstore.controller.CartController@4d409bb2

第二次使用Safari浏览器

com.reins.bookstore.serviceimpl.CartServiceImpl@2f097180

com.reins.bookstore.controller.CartController@4d409bb2

无状态

```
@Service
//@Scope("session")
public class CartServiceImpl implements CartService {
```

```
@RestController
//@Scope("session")
public class CartController {
    @Autowired
```

第一次使用Google chrome浏览器

com.reins.bookstore.serviceimpl.CartServiceImpl@6de9bba2

com.reins.bookstore.controller.CartController@7682bf66

第二次使用Google chrome浏览器

com.reins.bookstore.serviceimpl.CartServiceImpl@6de9bba2

com.reins.bookstore.controller.CartController@7682bf66

第一次使用Safari浏览器

com.reins.bookstore.serviceimpl.CartServiceImpl@6de9bba2

com.reins.bookstore.controller.CartController@7682bf66

第二次使用Safari浏览器

com.reins.bookstore.serviceimpl.CartServiceImpl@6de9bba2

com.reins.bookstore.controller.CartController@7682bf66

结论

无状态情况下，每一个请求都使用的是同一个controller和service。这样有多用户同时使用时会有冲突。

有状态情况下，可以看见不同的浏览器发请求（不同的session），它的controller 和 service 是不一样的，同时，如果是一个session，controller和service是一样的。这就保存了状态。

选择了将书籍增加到购物车addToCart这个函数作为有状态的设计，因为不同的客户加入购物车一定是不一样的，需要保存其状态。

Task2

2.编写一个JMS程序，用来接收并异步地处理订单，具体功能与现有E-Book中的下订单相同，具体流程为：

- 1) 用户通过Web前端下订单到服务器，你的服务器端程序 A(Controller) 应该先反馈给用户订单已接收；
- 2) A 将订单数据发送到“Order”队列；
- 3) 服务器端另一个程序 B(Service) 一直监听Order队列，一旦读到消息，就立刻进行处理，将数据写入数据库。

```
OrderController.java
@GetMapping("/addOrderFromUser")
public Order addOrderFromUser (@RequestParam(required = false) Integer user_id,
                                @RequestParam(required = false) Integer order_price,
                                @RequestParam(required = false) String date
) {
    JmsTemplate jmsTemplate = applicationContext.getBean(JmsTemplate.class);
    System.out.println("Sending an order message.");
    Order order = new Order(user_id, order_price, date);
    jmsTemplate.convertAndSend(destinationName: "orderBox", new Order(user_id, order_price, date));
    return order;
}
```

```
OrderReceiver.java
@Component
public class orderReceiver {
    @Autowired
    WebApplicationContext applicationContext;
    @Autowired
    private OrderService orderService;
    @Autowired
    private GetCartService getCartService;

    @JmsListener(destination = "orderBox", containerFactory = "myFactory")
    public void receiveMessage(Order order) {
        Order order1 = orderService.addOrderFromUser(order.getUserid(), order.getOrder_price(), order.getDate());
        List<Cart> cartList = getCartService.getCart();
        for(Cart c: cartList){
            System.out.println(order1.getOrderid() + " " + c.getBookId() + " " + c.getNumber());
            OrderItem item = orderService.addOrderItem(order1.getOrderid(), c.getBookId(), c.getNumber());
        }
        getCartService.clearCart();
        System.out.println("Received order < order_id:" + order1.getOrderid() + " user_id:" + order1.getUserid() + " cr
    }
}
```

流程为：前端发送订单信息，OrderController 作为发送者，发送一个new 的 order 到"orderBox"，随后接受者orderReceiver 读取信息，将order写入数据库并得到对应的order_id，从cart里面取出所有的购物车item，然后生成订单。然后清空购物车。

结果

	cart_id	author	name	number	price	book_id	user_id
▶	254	Ashlee Vance	Elon Musk	1	3299	4	1
	253	JK Rowling	Harry Potter	2	1899	2	1
	NULL	NULL	NULL	NULL	NULL	NULL	NULL

开始时购物车mysql

发送请求

`http://localhost:9090/addOrderFromUser?user_id=1&order_price=16994&date=2021-09-22`

```

Hibernate: delete from cart where cart_id=?
Hibernate: delete from cart where cart_id=?
Received order < order_id:803 user_id:1 order_price:16994 date:2021-09-22>

```

后端console

	57	4	1	803
	58	2	2	803
	NULL	NULL	NULL	NULL

结束时mysql: orderItem 加入了两条数据

	803	16994	1	2021-09-22
	NULL	NULL	NULL	NULL

结束时mysql: orders 加入了一条数据