

[今日课程大纲]

订单需求分析

订单系统实现

[知识点详解]

一. 订单需求分析

1. 在订单确认页面中从 redis 中把数据查询出来并显示

1.1 确认商品数量

2. 提交订单时复杂数据类型传递.

2.1 向 mysql 中 3 个表新增.

二. 实现订单确认页面功能

1. 在 ego-commons 中 TbltemChild 中添加新属性

```
/**  
 * 库存是否足  
 */  
  
private Boolean enough;
```

2. 新建 ego-order 项目,tomcat 端口 8086

3. 在 ego-order 中添加拦截器,实现登录拦截

4. 在 ego-order 中新建 TbOrderService 及实现类

```
public interface TbOrderService {

    /**
     * 确认订单信息包含的商品
     * @param id
     * @return
     */
    List<TbItemChild> showOrderCart(List<Long>
id,HttpServletRequest request);
}

@Service
public class TbOrderServiceImpl implements
TbOrderService {

    @Resource
    private JedisDao jedisDaoImpl;

    @Value("${cart.key}")
    private String cartKey;

    @Value("${passport.url}")
    private String passprtUrl;

    @Reference
    private TbItemDubboService tbItemDubboServiceImpl;

    @Override
```

```
public List< TbItemChild> showOrderCart(List<Long>
ids, HttpServletRequest request) {

    String token = CookieUtils.getCookieValue(request,
"TT_TOKEN");

    String result =
HttpClientUtil.doPost(passprtUrl+token);

    EgoResult er = JsonUtils.jsonToPojo(result,
EgoResult.class);

    String key =
cartKey+((LinkedHashMap)er.getData()).get("username")
;

    String json = jedisDaoImpl.get(key);

    List< TbItemChild> list =
JsonUtils.jsonToList(json, TbItemChild.class);

    List< TbItemChild> listNew = new ArrayList<>();

    for (TbItemChild child : list) {

        for (Long id : ids) {

            if((long)child.getId()==(long)id){

                //判断购买量是否大于等于库存

                TbItem tbItem =
tbItemDubboServiceImpl.selById(id);
```

```
        if(tbItem.getNum()>=child.getNum()){
            child.setEnough(true);
        }else{
            child.setEnough(false);
        }
        listNew.add(child);
    }

}

return listNew;
}

}
```

5. 在 ego-order 中新建控制器

```
@Controller

public class OrderController {

    @Resource

    private TbOrderService tbOrderServiceImpl;

    /**
```

```
* 显示确认页面

* @param model

* @param ids

* @param request

* @return

*/

@RequestMapping("order/order-cart.html")

public String showCartOrder(Model

model,@RequestParam("id") List<Long>

ids,HttpServletRequest request){

    model.addAttribute("cartList",

tbOrderServiceImpl.showOrderCart(ids, request));

    return "order-cart";

}

}
```

三.创建订单

1.在 ego-service 中新建接口和实现类

```
public interface TbOrderDubboService {

    /**

    * 创建订单

    * @param order
```

```
    * @param list
    * @param shipping
    * @return
    */

    int insOrder(TbOrder order, List<TbOrderItem>
list, TbOrderShipping shipping) throws Exception;
}

public class TbOrderDubboServiceImpl implements
TbOrderDubboService{

    @Resource

    private TbOrderMapper tbOrderMapper;

    @Resource

    private TbOrderItemMapper tbOrderItemMapper;

    @Resource

    private TbOrderShippingMapper tbOrderShippingMapper;

    @Override

    public int insOrder(TbOrder order, List<TbOrderItem>
list, TbOrderShipping shipping) throws Exception {

        int index = tbOrderMapper.insertSelective(order);

        for (TbOrderItem tbOrderItem : list) {

            index+=tbOrderItemMapper.insertSelective(tbOrderIte
```

```
m);  
  
    }  
  
    index+=tbOrderShippingMapper.insertSelective(shippi  
ng);  
  
    if(index==2+list.size()){  
        return 1;  
    }else{  
        throw new Exception("创建订单失败");  
    }  
}  
  
}
```

2.在 ego-service-impl 中新建接口注册

```
<!-- 订单 -->  
  
    <dubbo:service  
interface="com.ego.dubbo.service.TbOrderDubboService"  
ref="tbOrderDubboServiceImpl"></dubbo:service>  
  
    <bean                                id="tbOrderDubboServiceImpl"  
class="com.ego.dubbo.service.impl.TbOrderDubboService  
Impl"></bean>
```

3.在 ego-order 中 TbOrderService 中添加方法

```
/**
 * 创建订单
 * @param param
 * @return
 */
EgoResult create(MyOrderParam
param,HttpServletRequest request);

@Override
    public EgoResult create(MyOrderParam
param,HttpServletRequest request) {
        //订单表数据
        TbOrder order = new TbOrder();
        order.setPayment(param.getPayment());
        order.setPaymentType(param.getPaymentType());
        long id = IDUtils.genItemId();
        order.setOrderId(id+"");
        Date date =new Date();
        order.setCreateTime(date);
        order.setUpdateTime(date);
        String token = CookieUtils.getCookieValue(request,
"TT_TOKEN");
        String result =
```



```
HttpClientUtil.doPost(passportUrl+token);

    EgoResult er = JsonUtils.jsonToPojo(result,
EgoResult.class);

    Map user= (LinkedHashMap)er.getData();

    order.setUserId(Long.parseLong(user.get("id").toString()
ing()));

    order.setBuyerNick(user.get("username").toString())
;

    order.setBuyerRate(0);

    //订单-商品表
    for (TbOrderItem item : param.getOrderItems()) {
        item.setId(IDUtils.genItemId()+"");
        item.setOrderId(id+"");
    }

    //收货人信息
    TbOrderShipping shipping =
param.getOrderShipping();

    shipping.setOrderId(id+"");

    shipping.setCreated(date);

    shipping.setUpdated(date);
```

```
EgoResult erResult = new EgoResult();

try {

    int index = tbOrderDubboService.insOrder(order,
param.getOrderItems(), shipping);

    if(index>0){

        erResult.setStatus(200);

        //删除购买的商品

        String json =
jedisDaoImpl.get(cartKey+user.get("username"));

        List<TbItemChild> listCart =
JsonUtils.jsonToList(json, TbItemChild.class);

        List<TbItemChild> listNew = new
ArrayList<>();

        for (TbItemChild child : listCart) {

            for (TbOrderItem item :
param.getOrderItems()) {

                System.out.println("1"+child.getId().longValue());

                System.out.println("2"+Long.parseLong(item.getItemI
d()));
```

```
        if(child.getId().longValue()==Long.parseLong(item.get  
etItemId())){  
  
            listNew.add(child);  
  
        }  
    }  
}  
  
for (TbItemChild mynew : listNew) {  
    listCart.remove(mynew);  
}  
  
jedisDaoImpl.set(cartKey+user.get("username"),  
JsonUtils.objectToJson(listCart));  
  
    //删除  
}  
  
} catch (Exception e) {  
    e.printStackTrace();  
}  
  
return erResult;  
  
}
```

4.在 ego-order 控制器中添加

```
/**
 * 创建订单
 * @param param
 * @param request
 * @return
 */
@RequestMapping("order/create.html")
public String createOrder(MyOrderParam
param,HttpServletRequest request){
    EgoResult er = tbOrderServiceImpl.create(param,
request);
    if(er.getStatus()==200){
        return "my-orders";
    }else{
        request.setAttribute("message", "订单创建失败
");
        return "error/exception";
    }
}
```