

Федеральное государственное бюджетное образовательное учреждение  
высшего профессионального образования  
«Московский государственный технический университет имени Н. Э.  
Баумана»



Пример по  
выполнению  
Домашнего  
задания № 2

«Применение паттернов  
проектирования» по курсу  
Технологии разработки ПО

Составила Макрушина В.А.

Ван ЧаоЧао иу5и-12М

Москва – 2021

## **Цель работы**

Изучить основные паттерны проектирования, их особенности и область применения. Получить практические навыки программирования паттернов. Освоить технологию включения паттернов в собственную программу.

## **Порядок и время проведения работы**

Работа выполняется самостоятельно в часы внеаудиторных занятий. По итогам составляется и защищается отчет в бумажном виде, а также проводится демонстрация работающей программы.

## **Задание**

- 1) Разработать программу (основные прецеденты) на основе темы, выданной преподавателем (по варианту).
- 2) Реализовать в программе паттерны (по варианту) бизнес-логики и работы с БД.
- 3) Составить набор диаграмм классов и последовательностей, которые демонстрируют структуру и поведение программы.
- 4) Отдельно составить диаграммы классов и последовательностей для иллюстрации примененных паттернов.

### **К защите:**

#### **Программная реализация:**

- Работающая программа, реализующая основные функции системы (по варианту).
- Программа должна содержать реализацию паттернов проектирования (по варианту).

#### **В отчет:**

- 1) Диаграмма(ы) классов системы,
- 2) Диаграммы последовательностей для основных функций программы,
- 3) Диаграммы классов и последовательностей для иллюстрации примененных паттернов.

### **класс car**

stockNumber:int  
Supplier:string  
vehicleBrands:String  
whetherToSell:bool  
+depositInTheWarehouse() используется для Хранить на складе  
+moveOutOfTheWarehouse () используется для Забрать машину со склада  
+sold() для продавать

### **Inventory list(Список инвентаря) класс**

carsInStock:Car  
inventoryManager:inventorylistadd  
addTo(): Добавить информацию о заказе  
logOut(): Удалить заказ

### **staff member класс.**

Должность персонала

### **Supplier класс.**

idNumber: Код поставщиком  
phoneNumber: номер телефона

### **Sales Order класс.**

+carl:Car  
+customer: string :  
+numberOfCars:int:  
+salesMangementStaff:Salestaff

### **customer класс**

address: string  
id: int:  
name: String

## Диаграмма классов для прецедента создание билетов на концерт (определение классов сущностей )

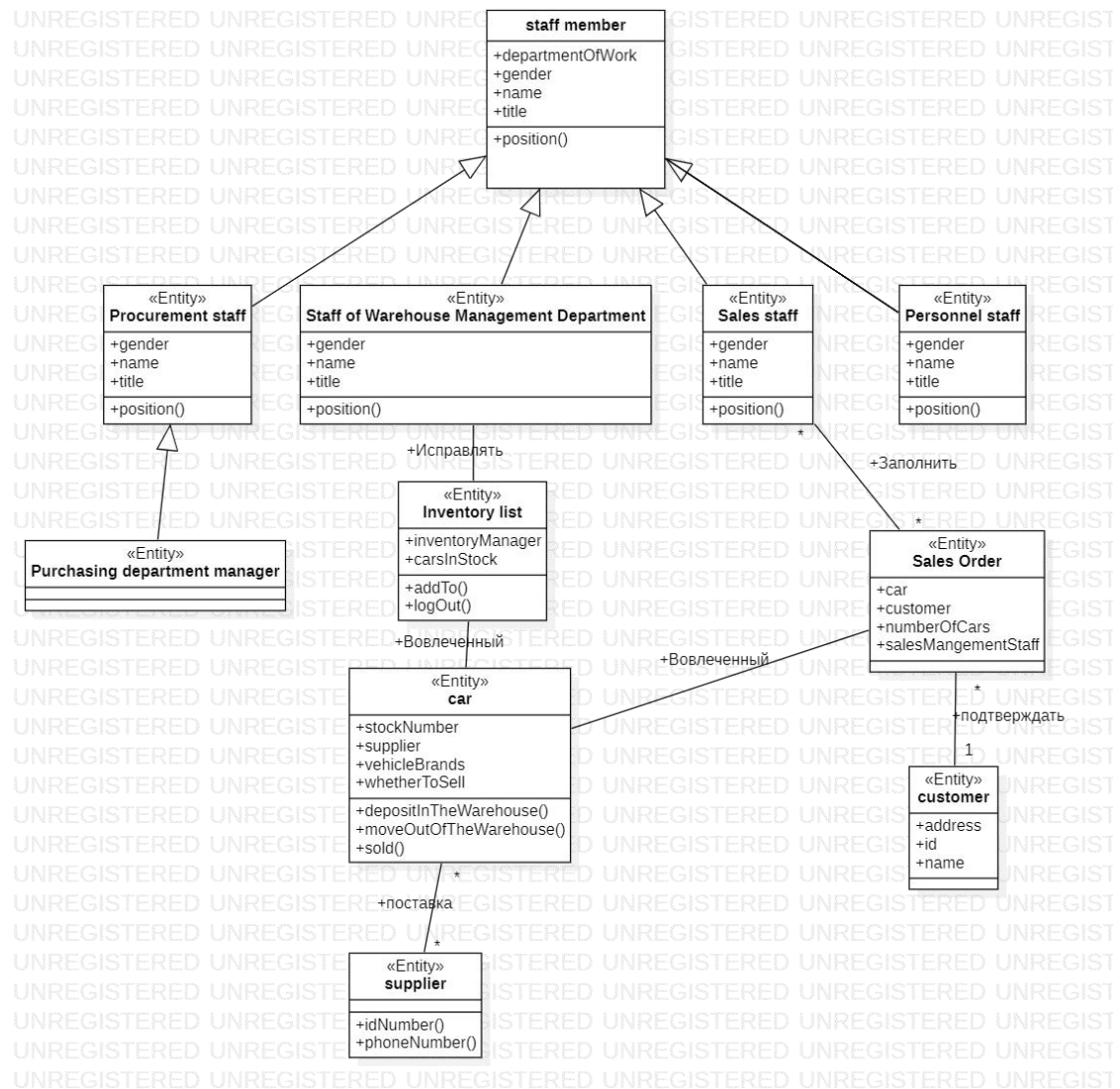


Рис.1 «Обзорная» диаграмма всех классов

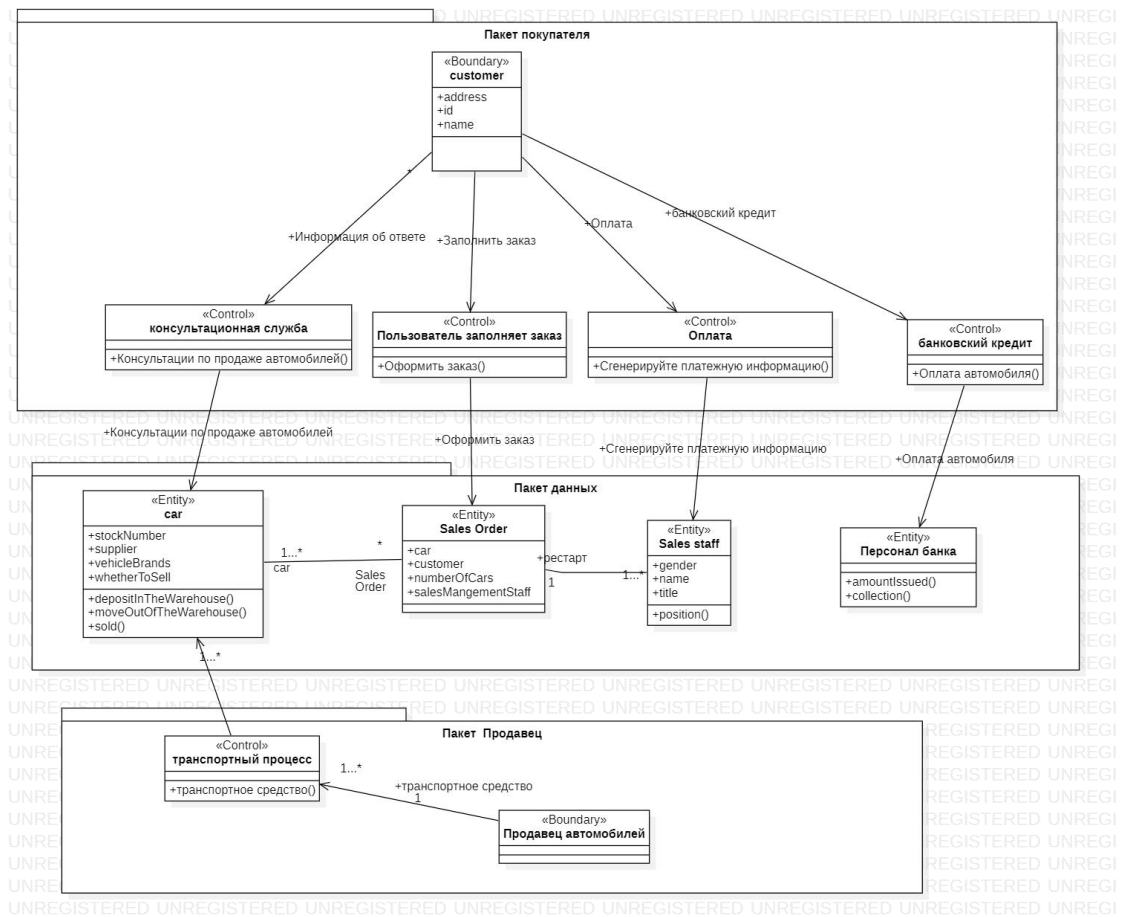


Рис. 2. «Обзорная» диаграмма пакетов ( классов)

## **Диаграмма последовательности для прецедента создание билетов на концерт (определить граничные и управляющие классы)**

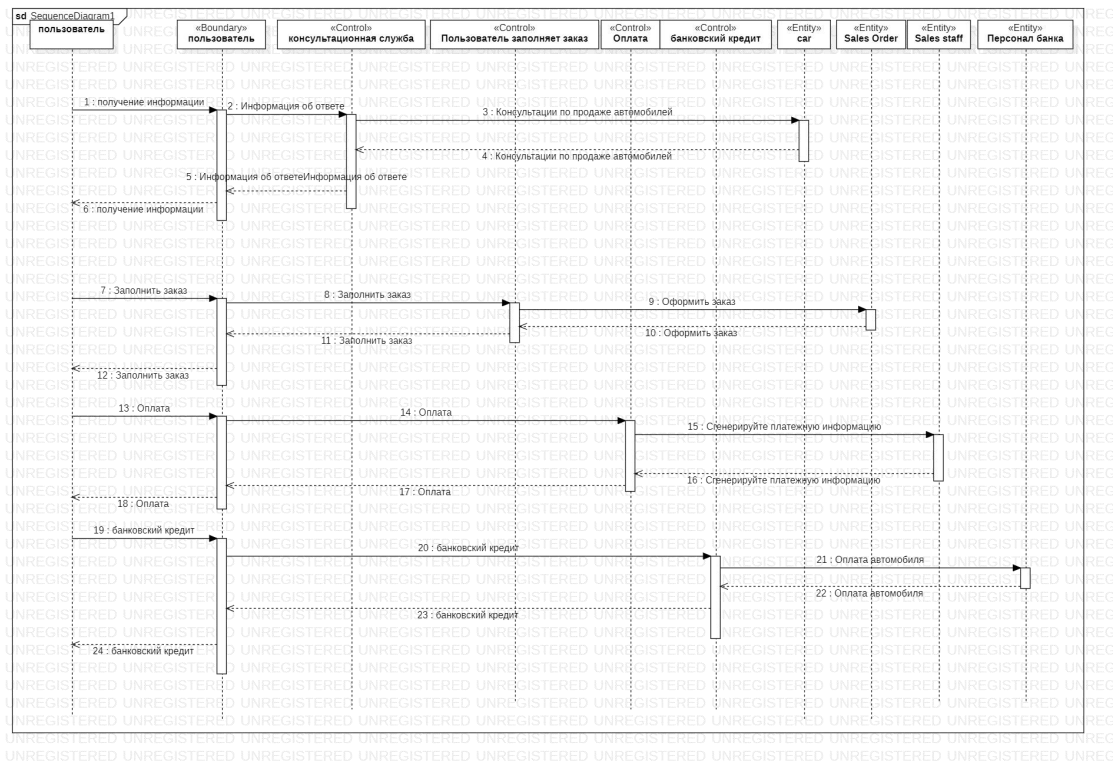


Рис.3. Диаграмма последовательностей для кооперации «Оформление заказа»

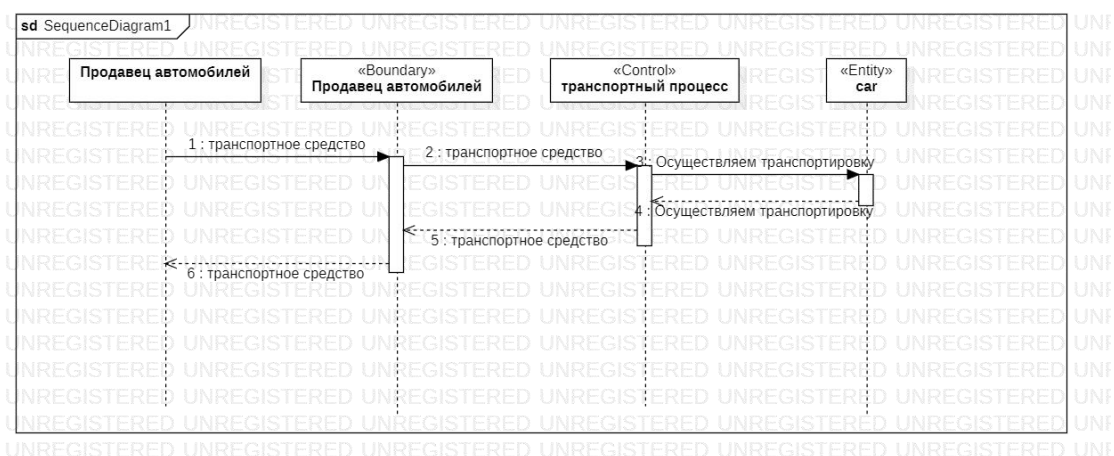


Рис.4. Диаграмма последовательностей для транспортный процесс

# Программный код с пояснениями


## 1.интерфейс входа

请登录 注册

消息 购物车 收藏夹

淘身边 首页 个人信息 我的订单

Search



用户名 Ван Чаочао

密码 .....

验证码  umcm

登录 忘记密码?

## 2.база данных mysql (таблица имен пользователей) при подключении Navicat

对象	user @sho...	imagepath...	comment ...	collection ...	chat
开始事务	文本	筛选	排序	导入	导出
数据生成	创建图表				
userId	username	password	regTime	email	telephone
1	Ван Чаочао	123456	2022-01-13 0	262385946	18801035010
2	maria	12345678	2022-01-15 1	262385946	18801035010

userId

类型 int(12)

不是 null

### 3.исходный код

```
package com.neu.shop.util.verificate;

import javax.imageio.ImageIO;
import java.awt.*;
import java.awt.image.BufferedImage;
import java.io.IOException;
import java.io.OutputStream;
import java.util.Random;

/**
 * Created by Ван Чаочао on 2021/12/22.
 */
public class Verificate {

    // 验证码图片中可以出现的字符集，可根据需要修改

    private char mapTable[] = { 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j',
        'k', 'l', 'm', 'n', 'o', 'p', 'q',
        'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', '0', '1', '2', '3', '4',
        '5', '6', '7', '8', '9' };

    /**
     * 功能: 生成彩色验证码图片 参数width 为生成图片的宽度, 参数height 为生成图片的高
     度, 参数为页面的输出流
     */
    public String getCertPic(int width, int height, OutputStream os) {
        if (width <= 0)
            width = 60;
        if (height <= 0)
            height = 20;
        BufferedImage image = new BufferedImage(width, height,
            BufferedImage.TYPE_INT_RGB);

        // 获取图形上下文

        Graphics g = image.getGraphics();

        // 设定背景色

        g.setColor(new Color(0x9FDCB1));
```



```
g.fillRect(0, 0, width, height);

// 画边框

g.setColor(new Color(0x9FDCB1));
g.drawRect(0, 0, width - 1, height - 1);

// 取随机产生的认证码

String strEnsure = "";

// 4 代表 4 位验证码,如果要生成更多位的认证码,则加大数值

for (int i = 0; i < 4; ++i) {
    strEnsure += mapTable[(int) (mapTable.length * Math.random())];
}

// 将认证码显示到图像中,如果要生成更多位的认证码,增加 drawString 语句

g.setColor(new Color(0x172D44));
g.setFont(new Font("Atlantic Inline", Font.PLAIN, 18));
String str = strEnsure.substring(0, 1);
g.drawString(str, 8, 17);
str = strEnsure.substring(1, 2);
g.drawString(str, 20, 15);
str = strEnsure.substring(2, 3);
g.drawString(str, 35, 18);
str = strEnsure.substring(3, 4);
g.drawString(str, 45, 15);

// 随机产生 10 个干扰点

Random rand = new Random();
for (int i = 0; i < 10; i++) {
    int x = rand.nextInt(width);
    int y = rand.nextInt(height);
    g.drawOval(x, y, 1, 1);
}

// 释放图形上下文

g.dispose();
try {
    // 输出图像到页面

    ImageIO.write(image, "JPEG", os);
} catch (IOException e) {
    return "";
}

return strEnsure;
```

```
}  
}
```

```
package com.neu.shop.controller.front;  
  
import com.github.pagehelper.PageHelper;  
import com.github.pagehelper.PageInfo;  
import com.neu.shop.pojo.*;  
import com.neu.shop.service.AddressService;  
import com.neu.shop.service.GoodsService;  
import com.neu.shop.service.OrderService;  
import com.neu.shop.service.UserService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RequestParam;  
import org.springframework.web.bind.annotation.ResponseBody;  
  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpSession;  
import java.util.ArrayList;  
import java.util.Date;  
import java.util.List;  
  
/**  
 * Created by Ван Чаочао on 2021/12/22.  
 */  
@Controller  
public class CustomerController {  
  
    @RequestMapping("/login")  
    public String loginView(){  
        return "login";  
    }  
  
    @Autowired  
    private UserService userService;  
  
    @RequestMapping("/register")  
    public String register(){  
        return "register";  
    }  
}
```

```

@RequestMapping("/registerresult")
public String registerResult(User user,Model registerResult){
    List<User> userList=new ArrayList<>();
    UserExample userExample=new UserExample();
    userExample.or().andUsernameLike(user.getUsername());
    userList=userService.selectByExample(userExample);
    if (!userList.isEmpty())
    {
        registerResult.addAttribute("errorMsg","用户名被占用");

        return "register";
    }
    else {
        Date RegTime=new Date();
        user.setRegtime(RegTime);
        userService.insertSelective(user);
        return "redirect:/login";
    }
}

@RequestMapping("/loginconfirm")
public String loginConfirm(User user,Model loginResult,HttpServletRequest
request,@RequestParam("confirmlogo") String confirmlogo){
    HttpSession session=request.getSession();
    String verificationCode = (String) session.getAttribute("certCode");
    if (!confirmlogo.equals(verificationCode))
    {
        loginResult.addAttribute("errorMsg","验证码错误");

        return "login";
    }

    List<User> userList=new ArrayList<User>();
    UserExample userExample=new UserExample();

    userExample.or().andUsernameEqualTo(user.getUsername()).andPasswordEqualTo(user.
getPassword());
    userList=userService.selectByExample(userExample);
    if (!userList.isEmpty())
    {
        session.setAttribute("user",userList.get(0));
        return "redirect:/main";
    }
}

```

```

        else {

            loginResult.addAttribute("errorMsg", "用户名与密码不匹配");

            return "login";
        }
    }

    @RequestMapping("/information")
    public String information(Model userModel, HttpServletRequest request){
        HttpSession session=request.getSession();
        User user;
        Integer userId;
        user=(User) session.getAttribute("user");
        if (user==null)
        {
            return "redirect:/login";
        }
        userId=user.getUserid();
        user=userService.selectByPrimaryKey(userId);
        userModel.addAttribute("user",user);
        return "information";
    }

    @RequestMapping("/saveInfo")
    @ResponseBody
    public Msg saveInfo(String name, String email, String
telephone, HttpServletRequest request){
        HttpSession session=request.getSession();
        UserExample userExample=new UserExample();
        User user,updateUser=new User();
        List<User> userList=new ArrayList<>();
        Integer userid;
        user=(User)session.getAttribute("user");
        userid= user.getUserid();
        userExample.or().andUsernameEqualTo(name);
        userList=userService.selectByExample(userExample);
        if (userList.isEmpty())
        {
            updateUser.setUserid(userid);
            updateUser.setUsername(name);
            updateUser.setEmail(email);
            updateUser.setTelephone(telephone);
            userService.updateByPrimaryKeySelective(updateUser);

            return Msg.success("更新成功");
        }
    }

```

```

    }

    else {return Msg.fail("更新失败");}

}

@Autowired
private AddressService addressService;

@RequestMapping("/info/address")
public String address(HttpServletRequest request, Model addressModel){
    HttpSession session=request.getSession();
    User user=(User)session.getAttribute("user");
    if (user==null)
    {
        return "redirect:/login";
    }
    AddressExample addressExample=new AddressExample();
    addressExample.or().andUserIdEqualTo(user.getUserId());
    List<Address>
addressList=addressService.getAllAddressByExample(addressExample);
    addressModel.addAttribute("addressList",addressList);
    return "address";
}

@RequestMapping("/saveAddr")
@ResponseBody
public Msg saveAddr(Address address){

    addressService.updateByPrimaryKeySelective(address);

    return Msg.success("修改成功");
}

@RequestMapping("/deleteAddr")
@ResponseBody
public Msg deleteAddr(Address address){
    addressService.deleteByPrimaryKey(address.getAddressid());

    return Msg.success("删除成功");
}

@RequestMapping("/insertAddr")
@ResponseBody
public Msg insertAddr(Address address,HttpServletRequest request){

```

```

        HttpSession session=request.getSession();
        User user=new User();
        user=(User) session.getAttribute("user");
        address.setUserid(user.getUserid());
        addressService.insertSelective(address);

        return Msg.success("添加成功");
    }

    @Autowired
    private OrderService orderService;

    @Autowired
    private GoodsService goodsService;

    @RequestMapping("/info/list")
    public String list(HttpServletRequest request, Model orderModel){

        HttpSession session=request.getSession();
        User user;
        user=(User)session.getAttribute("user");

        if (user==null)
        {
            return "redirect:/login";
        }

        OrderExample orderExample=new OrderExample();
        orderExample.or().andUseridEqualTo(user.getUserid());
        List<Order> orderList=orderService.selectOrderByExample(orderExample);
        orderModel.addAttribute("orderList",orderList);
        Order order;
        OrderItem orderItem;
        List<OrderItem> orderItemList=new ArrayList<>();
        Goods goods;
        Address address;
        for (Integer i=0;i<orderList.size();i++)
        {
            order=orderList.get(i);
            OrderItemExample orderItemExample=new OrderItemExample();
            orderItemExample.or().andOrderidEqualTo(order.getOrderid());
            orderItemList=orderService.getOrderItemByExample(orderItemExample);
            List<Goods> goodsList=new ArrayList<>();
            List<Integer> goodsIdList=new ArrayList<>();

```

```

        for (Integer j=0;j<orderItemList.size();j++)
        {
            orderItem=orderItemList.get(j);
            goodsIdList.add(orderItem.getGoodsid());
        }
        GoodsExample goodsExample=new GoodsExample();
        goodsExample.or().andGoodsidIn(goodsIdList);
        goodsList=goodsService.selectByExample(goodsExample);
        order.setGoodsInfo(goodsList);
        address=addressService.selectByPrimaryKey(order.getAddressid());
        order.setAddress(address);
        orderList.set(i,order);
    }

    orderModel.addAttribute("orderList",orderList);

    return "list";
}

/* @RequestMapping("/info/list")
public String list(HttpServletRequest request,Model orderModel,
                @RequestParam(value = "pageIssend",defaultValue = "1")
Integer pnIssend,
                @RequestParam(value = "pageIsrecive",defaultValue = "1")
Integer pnIsrecive,
                @RequestParam(value = "pageIscompelete",defaultValue = "1")
Integer pnIscompelete

){

    //一页显示几个数据

    PageHelper.startPage(pnIssend, 3);
    PageHelper.startPage(pnIsrecive, 3);
    PageHelper.startPage(pnIscompelete, 3);
    HttpSession session=request.getSession();
    User user;
    user=(User)session.getAttribute("user");

    if (user==null)
    {
        return "redirect:/login";
    }
}

```

```

OrderExample orderExample=new OrderExample();
orderExample.or().andUserIdEqualTo(user.getUserid());
List<Order> orderList=orderService.selectOrderByExample(orderExample);
/** orderModel.addAttribute("orderList",orderList);**/
Order order;
OrderItem orderItem;
List<OrderItem> orderItemList=new ArrayList<>();
Goods goods;
Address address;
for (Integer i=0;i<orderList.size();i++)
{
    order=orderList.get(i);
    OrderItemExample orderItemExample=new OrderItemExample();
    orderItemExample.or().andOrderidEqualTo(order.getOrderid());
    orderItemList=orderService.getOrderItemByExample(orderItemExample);
    List<Goods> goodsList=new ArrayList<>();
    List<Integer> goodsIdList=new ArrayList<>();
    for (Integer j=0;j<orderItemList.size();j++)
    {
        orderItem=orderItemList.get(j);
        goodsIdList.add(orderItem.getGoodsid());
    }
    GoodsExample goodsExample=new GoodsExample();
    goodsExample.or().andGoodsidIn(goodsIdList);
    goodsList=goodsService.selectByExample(goodsExample);
    order.setGoodsInfo(goodsList);
    address=addressService.selectByPrimaryKey(order.getAddressid());
    order.setAddress(address);
    orderList.set(i,order);
}

```

//显示几个页号

```
PageInfo pageIssend = new PageInfo(orderList,2);
```

```
PageInfo pageIsrecive = new PageInfo(orderList,2);
```

```
PageInfo pageIscompelete = new PageInfo(orderList,2);
```

```
orderModel.addAttribute("pageInfoIssend", pageIssend);
```



```

        orderModel.addAttribute("pageInfoIsrecive", pageIsrecive);

        orderModel.addAttribute("pageInfoIscompelete", pageIscompelete);

        return "list";
    }*/

    @RequestMapping("/deleteList")
    @ResponseBody
    public Msg deleteList(Order order){
        orderService.deleteById(order.getOrderid());

        return Msg.success("删除成功");
    }

    @RequestMapping("/info/favorite")
    public String showFavorite(@RequestParam(value = "page",defaultValue = "1")
Integer pn, HttpServletRequest request,Model model){
        HttpSession session=request.getSession();
        User user=(User)session.getAttribute("user");
        if (user == null) {
            return "redirect:/login";
        }

        //一页显示几个数据

        PageHelper.startPage(pn, 16);

        FavoriteExample favoriteExample = new FavoriteExample();
        favoriteExample.or().andUserIdEqualTo(user.getUserid());
        List<Favorite> favoriteList =
goodsService.selectFavByExample(favoriteExample);

        List<Integer> goodsIdList = new ArrayList<Integer>();
        for (Favorite tmp:favoriteList) {
            goodsIdList.add(tmp.getGoodsid());
        }

        GoodsExample goodsExample = new GoodsExample();
        List<Goods> goodsList = new ArrayList<>();
        if (!goodsIdList.isEmpty()) {
            goodsExample.or().andGoodsidIn(goodsIdList);
            goodsList = goodsService.selectByExample(goodsExample);

```

```

    }

    //获取图片地址

    for (int i = 0; i < goodsList.size(); i++) {
        Goods goods = goodsList.get(i);

        List<ImagePath> imagePathList =
goodsService.findImagePath(goods.getGoodsid());

        goods.setImagePaths(imagePathList);

        //判断是否收藏

        goods.setFav(true);

        goodsList.set(i, goods);
    }

    //显示几个页号

    PageInfo page = new PageInfo(goodsList,5);
    model.addAttribute("pageInfo", page);

    return "favorite";
}

@RequestMapping("/savePsw")
@ResponseBody
public Msg savePsw(String Psw,HttpServletRequest request)
{
    HttpSession session=request.getSession();
    User user=(User) session.getAttribute("user");
    user.setPassword(Psw);
    userService.updateByPrimaryKeySelective(user);

    return Msg.success("修改密码成功");
}

@RequestMapping("/finishList")
@ResponseBody
public Msg finishiList(Integer orderid){
    Order order=orderService.selectByPrimaryKey(orderid);
    order.setIsreceive(true);
}

```

```
        order.setIscomplete(true);
        orderService.updateOrderByKey(order);

        return Msg.success("完成订单成功");
    }

    @RequestMapping("/logout")
    public String logout(HttpServletRequest request){
        HttpSession session=request.getSession();
        session.removeAttribute("user");
        return "redirect:/login";
    }
}
```