商品销售平台方案设计

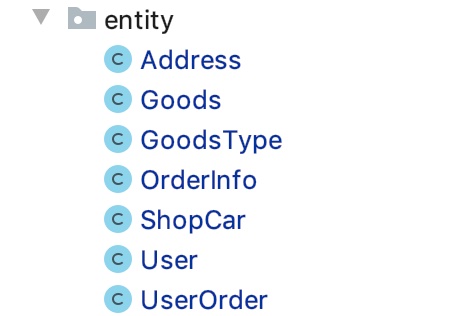
组长：杜一苇

组员：张悦

本次大作业设计的电子商城项目，具有以下功能：用户管理模块、订单管理模块、商品展示与管理模块、购物车模块、支付模块，积分模块，账户安全保护模块（通过对密码进行加密实现）。该文档仅列出相关接口设计，具体实现细节的相关配置（如数据库连接配置，mybatis等插件配置）、其他附加功能（如设备识别、分页显示）以及前端页面见项目代码。

方案设计如下：

1. pojo.entity层(实现了Serializable接口)



#### **User类**

成员变量：

private Integer userId; //用户id

private String nickName; //用户昵称

private String realName; //真实姓名

private String password; //密码

private Date lastLoginTime;//最后一次登陆时间

private String lastLoginIp; //最后一次登陆ip

private Integer points； //积分

private static final long serialVersionUID = 1L;

#### **Goods类**

成员变量：

private Integer goodsId; //用户id

private Integer goodsTypeId; //用户昵称

private String title; //真实姓名

private String img; //密码

private Double oldPrice;//原本价格

private Double price; //现价

private String onSale; //是否在售

private Integer count; //库存

private String merchantName; //商家名字

private Integer evaluationCount; //评价次数

private static final long serialVersionUID = 1L;

private String detail; //商品详情

相关方法：

public Integer getGoodsId() //一系列setter、getter方法

public void setGoodsId(Integer goodsId)

public Integer getGoodsTypeId()

public void setGoodsTypeId(Integer goodsTypeId)

public String getTitle()

public void setTitle(String title)

public String getImg()

public void setImg(String img)

public Double getOldPrice()

public void setOldPrice(Double oldPrice)

public Double getPrice()

public void setPrice(Double price)

public String getOnSale()

public void setOnSale(String onSale)

public Integer getCount()

public void setCount(Integer count)

public String getMerchantName()

public void setMerchantName(String merchantName)

public Integer getEvaluationCount()

public void setEvaluationCount(Integer evaluationCount)

public String getDetail()

public void setDetail(String detail)

#### **GoodsType类**

private String orderId; //订单id

private Integer userId; //用户id

private Double allCost; //订单总金额

private Date createTime; //下单时间

private Date payTime; //付款时间

private String receiverPhone; //收货人的电话

private String receiverName; // 收货人姓名

private String receiveAddress; //收获地址

private String payType; //付款方式：支付宝或者微信，银行卡

private String status; //订单状态：待付款，待收货，待评价，已退款，已取消

private Double actualPayment; //实际付款金额

private Double totalDeduction; //总减免

private static final long serialVersionUID = 1L;

相关方法：

public String getOrderId() //一系列setter、getter方法

public void setOrderId(String orderId)

public Integer getUserId()

public void setUserId(Integer userId)

public Double getAllCost() {

public void setAllCost(Double allCost) {

public Date getCreateTime() {

public void setCreateTime(Date createTime) {

public Date getPayTime() {

public void setPayTime(Date payTime) {

public String getReceiverPhone() {

public void setReceiverPhone(String receiverPhone) {

public String getReceiverName() {

public void setReceiverName(String receiverName) {

public String getReceiveAddress() {

public void setReceiveAddress(String receiveAddress) {

public String getPayType() {

public void setPayType(String payType) {

public String getStatus() {

public void setStatus(String status) {

public Double getActualPayment() {

public void setActualPayment(Double actualPayment) {

public Double getTotalDeduction() {

public void setTotalDeduction(Double totalDeduction) {

#### **Address类**

成员变量：

private Integer addressId; //收货地址id

private Integer userId; //外键

private String province; //省

private String city; //市

private String area; //区

private String detailAddress; //详细地址

private String phone; //电话

private String receiver; //收货人

private String moren; //是否设置为默认地址

private String tag; //地址标签：家、学校、公司

private static final long serialVersionUID = 1L;

相关方法：

public Integer getAddressId() //一系列setter、getter方法

public void setAddressId(Integer addressId)

public Integer getUserId()

public void setUserId(Integer userId)

public String getProvince()

public void setProvince(String province)

public String getCity()

public void setCity(String city)

public String getArea()

public void setArea(String area)

public String getDetailAddress()

public void setDetailAddress(String detailAddress)

public String getPhone()

public void setPhone(String phone)

public String getReceiver()

public void setReceiver(String receiver)

public String getMoren()

public void setMoren(String moren)

public String getTag()

public void setTag(String tag)

#### **UserOrder类**

成员变量：

private String orderId; //订单id

private Integer userId; //外键

private Double allCost; //订单总金额

private Date createTime;//下单时间

private Date payTime; //付款时间

private String receiverPhone; //收货人的电话

private String receiverName; //收货人姓名

private String receiveAddress; //收货地址

private String payType; //付款方式：支付宝或者微信，银行卡

private String status; //订单状态：待付款，待收货，待评价，已退款，已取消

private Double actualPayment; //实际付款金额

private Double totalDeduction; //总减免

private static final long serialVersionUID = 1L;

相关方法：

public String getOrderId() //一系列setter、getter方法

……

#### **OrderInfo类**

成员变量：

private String orderId; //订单id

private Integer userId; //外键

private Double allCost; //订单总金额

private Date createTime; //下单时间

private Date payTime; //付款时间

private String receiverPhone; //收货人的电话

private String receiverName; //收货人姓名

private String receiveAddress; //收获地址

private String payType; //付款方式：支付宝或者微信，银行卡

private String status; //订单状态：待付款，待收货，待评价，已退款，已取消

private Double actualPayment; //实际付款金额

private Double totalDeduction; //总减免

private static final long serialVersionUID = 1L;

相关方法：

public String getOrderId() //一系列setter、getter方法

……

#### **ShopCar类**

成员变量：

private Integer shopCarId; //购物车id

private Integer userId; //外键

private Integer goodsId; //商品id

private Integer count; //数量

private static final long serialVersionUID = 1L;

相关方法：

public Integer getShopCarId() //一系列setter、getter方法

……

1. pojo.dto层

DTO是展示层和服务层之间的数据传输对象。

**ResponseDTO类**

成员变量：

private String msg;//服务器返回给客户端的消息

private Boolean res;//返回的结果

private Object data;//返回的数据

private Integer status = 200;//返回的状态

private Integer errorCode;//错误代码

相关方法：

public ResponseDTO(String msg, Boolean res, Object data, Integer status, Integer

errorCode) //构造函数

public static ResponseDTO ok(String msg) //返回简单成功消息，只返回成功的消息

public static ResponseDTO ok(String msg, Object data)//返回简单成功消息，只返回成功

的消息，并且返回data数据

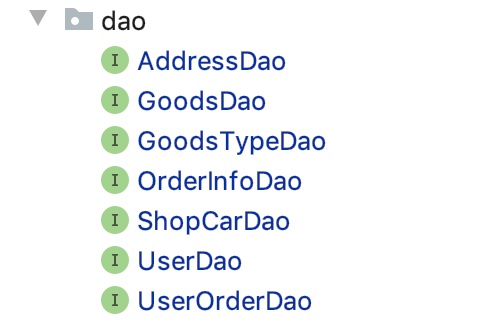
public static ResponseDTO fail(String msg, Object data, Integer status, Integer

errorCode)//返回失败消息，并返回data数据

public static ResponseDTO fail(String msg)//返回失败消息

1. dao层

设计dao层涉及到的接口，之后使用mybatis自动生成代码（在mybatis.mapper中查看相关代码）。

#### **UserDAO接口**

int deleteByPrimaryKey(Integer userId); //根据主键删除

int insert(User record); //插入数据

int insertSelective(User record); //选择性插入

User selectByPrimaryKey(Integer userId); //根据主键查询

int updateByPrimaryKeySelective(User record); //根据主键选择性的更新

int updateByPrimaryKey(User record); //更新全部

User selectUserByRealName(String realName); //根据真实名称查询用户信息

User selectByPhone(String phone); //根据电话号码查询用户基本信息

#### **GoodsDao接口**

int deleteByPrimaryKey(Integer goodsId); /根据主键删除

int insert(Goods record); //插入数据

int insertSelective(Goods record); //选择性插入

Goods selectByPrimaryKey(Integer goodsId); //根据主键查询

int updateByPrimaryKeySelective(Goods record);//根据主键选择性的更新

int updateByPrimaryKey(Goods record); //更新全部

List<Goods> getGoodsByTypeId(Integer typeId); //获得商品类型id

List<Goods> selectGoodsByIds(@Param("ids") List carVOS); //根据商品id查询

#### **GoodsTypeDao接口**

int deleteByPrimaryKey(Integer goodsTypeId); //根据商品类型删除

int insert(GoodsType record); //插入数据

int insertSelective(GoodsType record); //选择性插入

GoodsType selectByPrimaryKey(Integer goodsTypeId);//根据主键查询

int updateByPrimaryKeySelective(GoodsType record); //根据主键选择性的更新

int updateByPrimaryKey(GoodsType record); // 根据主键更新

List<GoodsTypeVO> selectTypesByParentId(int i); //根据商品父类型查询，获得列表

#### **AddressDao接口**

int deleteByPrimaryKey(Integer addressId); //根据主键删除地址

int insert(Address record); //插入新地址

int insertSelective(Address record); //选择性插入

Address selectByPrimaryKey(Integer addressId); //根据主键查询

int updateByPrimaryKeySelective(Address record); //根据主键选择性的更新

int updateByPrimaryKey(Address record); //根据主键更新

Address selectByUserId(@Param("userId") Integer userId, @Param("moren") String moren); //根据用户id查询地址

List<Address> getUserAddressList(Integer userId); //获得用户地址列表

#### **OrderInfoDao接口**

int deleteByPrimaryKey(Integer orderInfoId); //根据主键删除订单

int insert(OrderInfo record); //插入数据

int insertSelective(OrderInfo record); //选择性插入

OrderInfo selectByPrimaryKey(Integer orderInfoId);//根据主键查询

int updateByPrimaryKeySelective(OrderInfo record);//根据主键选择性的更新

int updateByPrimaryKey(OrderInfo record); //根据主键更新

#### **UserOrderDao接口**

int deleteByPrimaryKey(String orderId); //根据主键删除订单

int insert(UserOrder record); //插入数据

int insertSelective(UserOrder record); //选择性插入

UserOrder selectByPrimaryKey(String orderId); //根据主键查询

int updateByPrimaryKeySelective(UserOrder record);//根据主键选择性的更新

int updateByPrimaryKey(UserOrder record); //根据主键更新

#### **ShopCarDao接口**

int deleteByPrimaryKey(Integer shopCarId); //根据主键删除订单

int insert(ShopCar record); //插入数据

int insertSelective(ShopCar record); //选择性插入

ShopCar selectByPrimaryKey(Integer shopCarId); //根据主键查询

int updateByPrimaryKeySelective(ShopCar record);//根据主键选择性的更新

int updateByPrimaryKey(ShopCar record); //根据主键更新

ShopCar findCarByUserIdAndGoods(ShopCar shopCar); //根据用户id查找购物车

Integer selectCarCountByUserId(Integer userId); //根据商品id查找购物车

List<ShopCarVO> findUserCars(Integer userId); //根据用户id查询其购物车

1. 数据库

#### **user表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 字段类型 | 是否为主键 | 是否可为空 | 描述 |
| userId | Int(11) | YES | NO | 用户id |
| nick\_name | varchar(255) |  | YES | 用户昵称 |
| real\_name | varchar(550) |  | YES | 真实姓名 |
| phone | Varchar(255) |  | YES | 联系方式 |
| password | varchar(255) |  | YES | 账户密码 |
| last\_login\_time | varchar(255) |  | YES | 最后登陆时间 |
| last\_login\_ip | varchar(255) |  | YES | 最后登陆ip |
| points | Int(11) |  | YES | 积分 |

#### **goods表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 字段类型 | 是否为主键 | 是否可为空 | 描述 |
| goods\_id | Int(11) | YES | NO | 商品id |
| title | varchar(255) |  | YES | 商品标题 |
| img | varchar(255) |  | YES | 图片 |
| goods\_type\_id | varchar(255) |  | YES | 商品类型id |
| old\_price | varchar(255) |  | YES | 原价 |
| price | varchar(255) |  | YES | 现价 |
| on\_sale | varchar(255) |  | YES | 是否在售 |
| detail | mediumtext |  | YES | 商品详情 |
| count | Int(11) |  | YES | 存库 |

#### **address表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 字段类型 | 是否为主键 | 是否可为空 | 描述 |
| address\_id | Int(11) | YES | NO | 地址id |
| user\_id | varchar(255) |  | YES | 外键：用户id |
| province | varchar(550) |  | YES | 省 |
| city | Varchar(255) |  | YES | 市 |
| area | varchar(255) |  | YES | 区 |
| detailAddress | varchar(255) |  | YES | 详细地址 |
| phone | varchar(255) |  | YES | 收货电话 |
| receiver | varchar(255) |  | YES | 收货人 |
| moren | varchar(255) |  | YES | 是否为默认地址 |
| tag | varchar(255) |  | YES | 地址标签 |

#### **goods\_type表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 字段类型 | 是否为主键 | 是否可为空 | 描述 |
| goods\_type\_id | Int(11) | YES | NO | 商品类型的id |
| Parent\_type\_id | Int(11) |  | YES | 商品父类型的id |
| Type\_name | varchar(550) |  | YES | 类型名称 |
| sort | Int(11) |  | YES | 类型的序号 |
| Icon\_class | varchar(255) |  | YES | 图标样式 |

#### **user\_order表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 字段类型 | 是否为主键 | 是否可为空 | 描述 |
| order\_id | varchar(255) | YES | NO | 订单id |
| user\_id | Int(11) |  | YES | 外键：用户id |
| all\_cost | double |  | YES | 订单总金额 |
| create\_time | datetime |  | YES | 下单时间 |
| pay\_time | datetime |  | YES | 付款时间 |
| receiver\_phone | varchar(255) |  | YES | 收货人电话 |
| receiver\_name | varchar(255) |  | YES | 收货人姓名 |
| receiver\_address | varchar(255) |  | YES | 收货地址 |
| pay\_type | varchar(255) |  | YES | 付款方式 |
| status | varchar(255) |  | YES | 订单状态 |
| actual\_payment | double |  | YES | 实际付款金额 |
| Total\_decution | double |  | YES | 总减免 |

#### **shop\_car表**

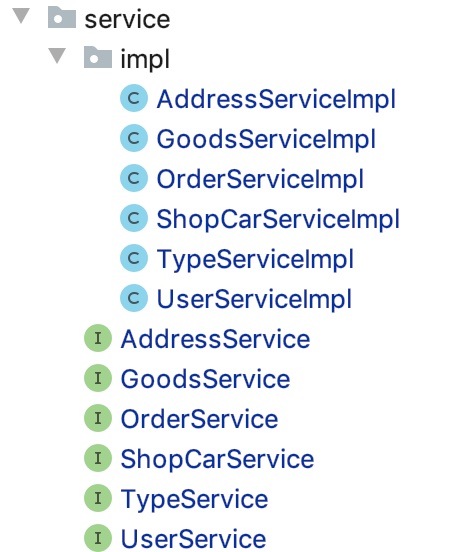
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 字段类型 | 是否为主键 | 是否可为空 | 描述 |
| shop\_car\_id | Int(11) | YES | NO | 购物车id |
| user\_id | Int(11) |  | YES | 用户id |
| goods\_id | Int(11) |  | YES | 商品id |
| count | Int(11) |  | YES | 数量 |

#### **order\_info表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 字段类型 | 是否为主键 | 是否可为空 | 描述 |
| order\_info\_id | Int(11) | YES | NO | 订单详情id |
| order\_id | varchar(255) |  | YES | 外键：订单id |
| orgin\_good\_id | Int(11) |  | YES | 原商品的连接 |
| price | double |  | YES | 下单时的商品价格 |
| count | Int(11) |  | YES | 下单数量 |
| img | varchar(255) |  | YES | 图片 |
| old\_price | double |  | YES | 商品原价 |

1. service层及其实现

Service层是建立在DAO层之上的，建立了DAO层后才可以建立Service层。每个模型都有一个Service接口，Service接口们实现业务操作，ServiceImp类实现Service接口，并且使用@Service注解交给springIOC(容器管理)后边就可以直接自动装配使用。



#### **5.1用户操作**

UserService接口 //关于用户的一些业务操作

*@param user*

*@return*

boolean login(User user); //用户登录，如果登录成功，返回true，登录失败返回false

void register(User user); //用户注册

UserServiceImpl类 //实现UserService接口

*@Resource*

*UserDao userDao;*

*@Override*

*public boolean login(User user)*

*@Override*

*public void register(User user)*

#### **5.2地址管理**

AddressService接口 //关于地址的业务操作

*@param userId*

*@return*

*List<Address> getUserAddressList(Integer userId); //查询用户所有收货地址，以供用户进行选择*

*Address getDefaultAddress(Integer userId); //获取默认地址*

AddressServiceImpl类 //实现AddressService接口

*@Resource*

*AddressDao addressDao;*

*@Override*

*public List<Address> getUserAddressList(Integer userId)*

*@Override*

*public Address getDefaultAddress(Integer userId)*

#### **5.3订单管理**

OrderService接口 //关于订单的业务操作

*@Transactional(propagation = Propagation.REQUIRED)*

*// 开启事务控制*

*ResponseDTO addOrder(UserOrder userOrder, List<ShopCarVO> carVOS); //添加订单*

*UserOrder getOrderDetailByOrderId(String orderId); //通过订单id获取订单详情*

*@param dbOrder*

*@return*

*boolean updatePayStatus(UserOrder dbOrder); //更新订单支付状态*

OrderServiceImpl类 //实现OrderService接口

*@Resource*

*UserOrderDao userOrderDao;*

*@Resource*

*OrderInfoDao orderInfoDao;*

*@Override*

*public ResponseDTO addOrder(UserOrder userOrder, List<ShopCarVO> carVOS)*

*@Override*

*public UserOrder getOrderDetailByOrderId(String orderId)*

*@Override*

*public boolean updatePayStatus(UserOrder dbOrder)*

#### **5.4商品管理**

GoodsService接口 //关于商品的一些业务操作

*@param goods*

*@return*

*Map<String, Object> add(Goods goods); //增加商品的功能，返回res结果true或者false*

*@param typeId*

*@return*

*ResponseDTO getGoodsByTypeId(Integer typeId);//根据商品类型寻找这个类型的所有商品*

*@param goodsId*

*@return*

*Goods getGoodsDetail(Integer goodsId); //根据商品id查询商品*

GoodsServiceImpl类 //实现GoodsService接口

*@Resource*

*GoodsDao goodsDao;*

*@Override*

*public Map<String, Object> add(Goods goods)*

*@Override*

*public ResponseDTO getGoodsByTypeId(Integer typeId)*

*@Override*

*public Goods getGoodsDetail(Integer goodsId)*

TypeService接口 //关于商品类型的业务操作

*@param goodsType*

*@return*

*ResponseDTO add(GoodsType goodsType) //增加商品类型户*

*@return*

*List<GoodsTypeVO> selectTypesByParentId(int i); //查询出所有一级商品类型*

TypeServiceImpl类 //实现TypeService接口

*@Resource*

*GoodsTypeDao goodsTypeDao;*

*@Override*

*public ResponseDTO add(GoodsType goodsType)*

*@Override*

*public List<GoodsTypeVO> selectTypesByParentId(int i)*

#### **5.5购物车**

ShopCarService接口 //关于购物车的一些业务操作

*ResponseDTO edit(ShopCar shopCar); //编辑购物车*

*@param userId*

*@return*

*Integer getCarCountByUserId(Integer userId); //根据用户id寻找他的购物车的数量*

*@param userId*

*@return*

*List<ShopCarVO> findUserCars(Integer userId); //根据用户id查询购物车列表*

ShopCarServiceImpl类 //实现ShopCarService接口

*@Resource*

*ShopCarDao shopCarDao;*

*@Resource*

*GoodsDao goodsDao;*

*@Override*

*public ResponseDTO edit(ShopCar shopCar)*

*@Override*

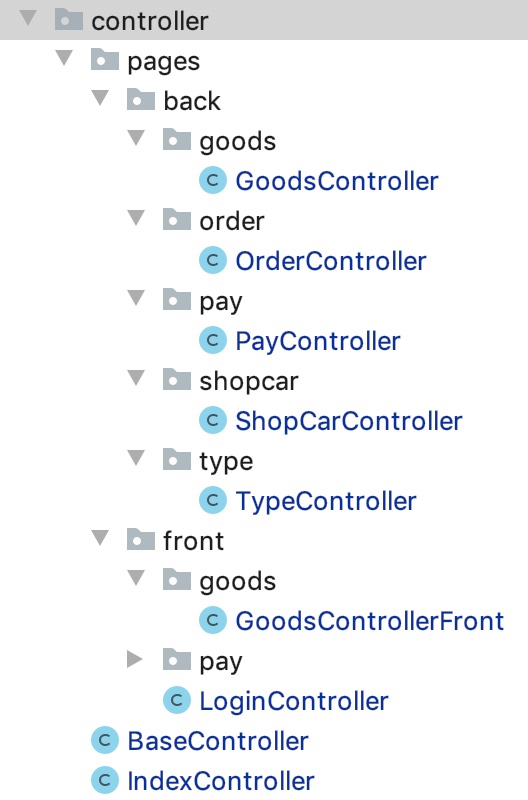
*public Integer getCarCountByUserId(Integer userId)*

*@Override*

*public List<ShopCarVO> findUserCars(Integer userId)*

1. controller层

controller层负责具体的业务模块流程的控制，在此层要调用service层的接口来控制业务流程具体功能实现。controller设计框架如下，分为前台（front）和后台（back）。



**BaseController** //封装一些控制器公用的方法，其他控制器继承此类

*//上传文件*

*@param uploadDir 以/开头，并且以/结尾*

*@param f*

*@return*

*protected String uploadFile(String uploadDir, MultipartFile f)*

*//获取Request对象*

*@return*

*protected HttpServletRequest getRequest()*

*//获取Session对象*

*@return*

*protected HttpSession getSession()*

*//获取 userid*

*@return*

*protected Integer getUserId()*

**前台（front）**

**LoginController**

*@Resource*

*UserService userService;*

*@Resource*

*ShopCarService shopCarService;*

*//跳转到登陆页面*

*@RequestMapping("loginPage")*

*String loginPage()*

*//验证用户名密码执行登录操作的方法*

*@return*

*@RequestMapping("login")*

*String login(User user, Model model, HttpSession session)*

*//页面跳转*

*@RequestMapping("toPage")*

*String toPage(HttpServletRequest request)*

*//注册*

*@RequestMapping("register")*

*String regist(User user)*

**GoodsControllerFront** （继承BaseController）

*@Resource*

*GoodsService goodsService;*

*//获得商品详情*

*@RequestMapping("goodsDetail/{goodsId}")*

*String goodsDetail(@PathVariable Integer goodsId, Model model)*

**PayControllerFront**

*@Resource*

*OrderService orderService;*

*//支付宝通知的接口*

*@RequestMapping("/aliPayResTb")*

*String aliPayResTb()*

*@RequestMapping("/aliPayResYb")*

*@ResponseBody*

*String aliPayResYb(HttpServletRequest request)*

*//确认支付*

*private String checkSignAndUpdateOrder(HttpServletRequest request)*

**后台（back）**

**GoodsController** （继承BaseController）

*@Resource*

*GoodsService goodsService;*

*@Resource*

*TypeService typeService;*

*//添加商品*

*@RequestMapping("addPre")*

*String addPre(Model model)*

*@return*

*@RequestMapping("add")*

*@ResponseBody*

*Map<String, Object> add(Goods goods, MultipartFile pic)*

*@return*

*@RequestMapping("getGoodsByTypeId/{typeId}")*

*@ResponseBody*

*ResponseDTO getGoodsByTypeId(@PathVariable Integer typeId)*

**OrderController** （继承BaseController）

*@Resource*

*AddressService addressService;*

*@Resource*

*OrderService orderService;*

*//添加订单*

*@RequestMapping("addPre")*

*String addPre(String orderInfoStr, Model model, Double allCost)*

*@RequestMapping("add")*

*String add(UserOrder userOrder, String orderInfos, Model model)*

**PayController**

*@Resource*

*OrderService orderService;*

*//阿里电脑扫码支付的表单回传到页面*

*@RequestMapping("aliWapPay")*

*@ResponseBody*

*Object aliWapPay(HttpServletRequest request)*

*//向支付宝/微信发起请求，并将直接跳转至支付宝/微信支付页面*

*@RequestMapping("/unifiedPay")*

*String unifiedPay(UserOrder order, HttpServletRequest request,*

*@RequestHeader("user-agent") String userAgent)*

**ShopCarController**（继承BaseController）

*@Resource*

*ShopCarService shopCarService;*

*//编辑购物车*

*@return*

*@RequestMapping("edit")*

*@ResponseBody*

*ResponseDTO edit(ShopCar shopCar, HttpSession session)*

*//查询并展示购物车*

*@RequestMapping("shwoMyCar")*

*String shwoMyCar(Model model)*

1. 账户安全保护

**PasswordDecoderUtil类** //实现账户密码加密

*//准备盐值*

*public static final String SALT = "TIEI\_DP";*

*//获得用户传到controller的密码，此时的密码是明文*

*@param password*

*@return*

*//使用MD5算法实现密码加密，得到密文*

*public static String getDecodePassword(String password)*