

# 第11章 订单

## 课程内容

- 1) 完成订单结算页渲染
- 2) 完成用户下单实现
- 3) 完成库存变更实现

## 1 订单结算页

### 1.1 收件地址分析

用户从购物车页面点击结算，跳转到订单结算页，结算页需要加载用户对应的收件地址，如下图：

收件人信息	
李煜	北京市海淀区三环内中关村软件园9号楼 15932223201 默认地址
李西西	北京市昌平区建材城西路金燕龙办公楼 18545692564
王希	河北省衡水市大庆西路888号 18758946254

表结构分析：

```
CREATE TABLE `tb_address` (  
  `id` int(11) NOT NULL AUTO_INCREMENT,  
  `username` varchar(50) DEFAULT NULL COMMENT '用户名',  
  `provinceid` varchar(20) DEFAULT NULL COMMENT '省',  
  `cityid` varchar(20) DEFAULT NULL COMMENT '市',  
  `areaid` varchar(20) DEFAULT NULL COMMENT '县/区',  
  `phone` varchar(20) DEFAULT NULL COMMENT '电话',  
  `address` varchar(200) DEFAULT NULL COMMENT '详细地址',  
  `contact` varchar(50) DEFAULT NULL COMMENT '联系人',  
  `is_default` varchar(1) DEFAULT NULL COMMENT '是否是默认 1默认 0否',  
  `alias` varchar(50) DEFAULT NULL COMMENT '别名',  
  PRIMARY KEY (`id`)  
) ENGINE=InnoDB AUTO_INCREMENT=66 DEFAULT CHARSET=utf8;
```

我们可以根据用户登录名去tb\_address表中查询对应的数据。

### 1.2 实现用户收件地址查询

#### 1.2.1 代码实现

(1)业务层

业务层接口

修改changgou-service-user微服务，需改com.changgou.user.service.AddressService接口，添加根据用户名查询用户收件地址信息，代码如下：

北京市昌平区建材城西路金燕龙办公楼一层 电话：400-618-9090



```
/* 收件地址查询 */
* @param username
* @return
*/
List<Address> list(String username);
```

### 业务层接口实现类

修改changgou-service-user微服务，修改com.changgou.user.service.impl.AddressServiceImpl类，添加根据用户名查询用户收件地址信息实现方法，如下代码：

```
/**
 * 收件地址查询
 * @param username
 * @return
 */
@Override
public List<Address> list(String username) {
    Address address = new Address();
    address.setUsername(username);
    return addressMapper.select(address);
}
```

### (2)控制层

修改changgou-service-user微服务，修改com.changgou.user.controller.AddressController，添加根据用户名查询用户收件信息方法，代码如下：

```
@Autowired
private TokenDecode tokenDecode;

/**
 * 用户收件地址
 */
@GetMapping(value = "/list")
public Result<List<Address>> list(){
    //获取用户登录信息
    Map<String, String> userMap = tokenDecode.getUserInfo();
    String username = userMap.get("username");
    //查询用户收件地址
    List<Address> addressList = addressService.list(username);
    return new Result(true, StatusCode.OK, "查询成功!", addressList);
}
```

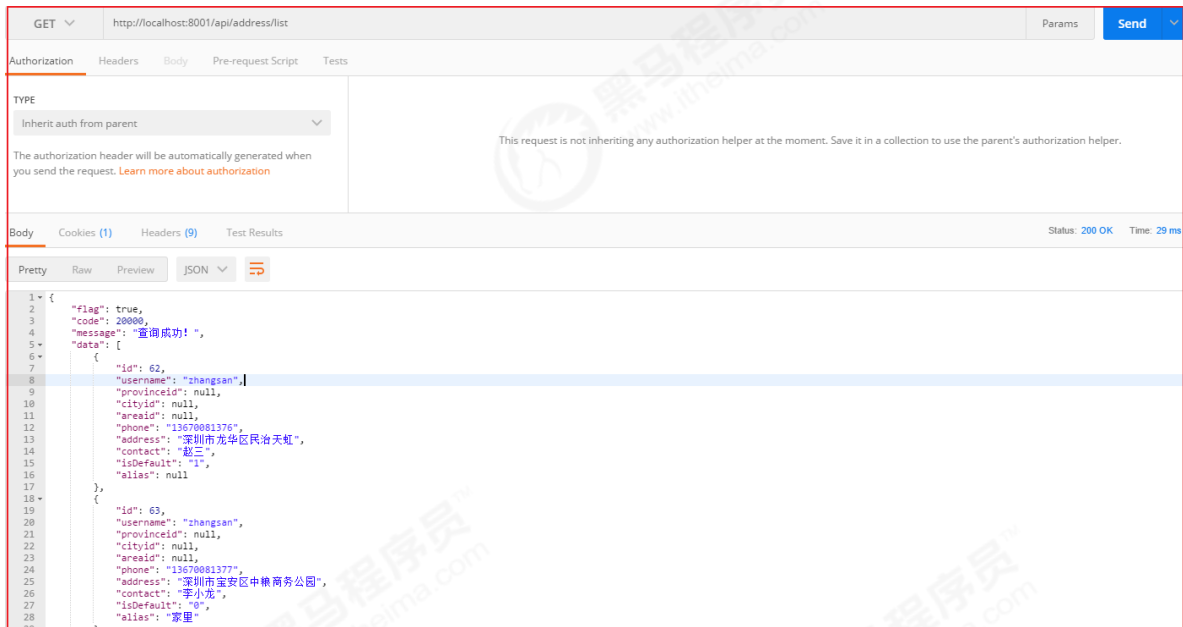
### (3)创建TokenDecode

在changgou-user-service的com.changgou.UserApplication中创建TokenDecode,代码如下：

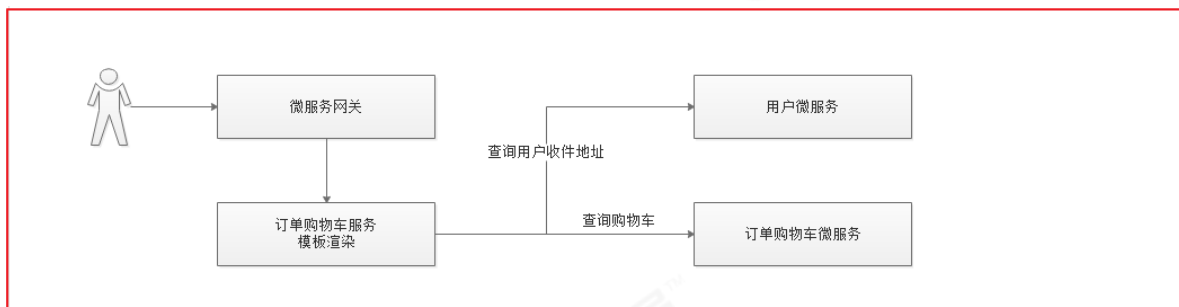
```
return new TokenDecode();  
}
```

## 1.2.2 测试

Postman访问 <http://localhost:8001/api/address/list>



## 1.3 页面模板渲染

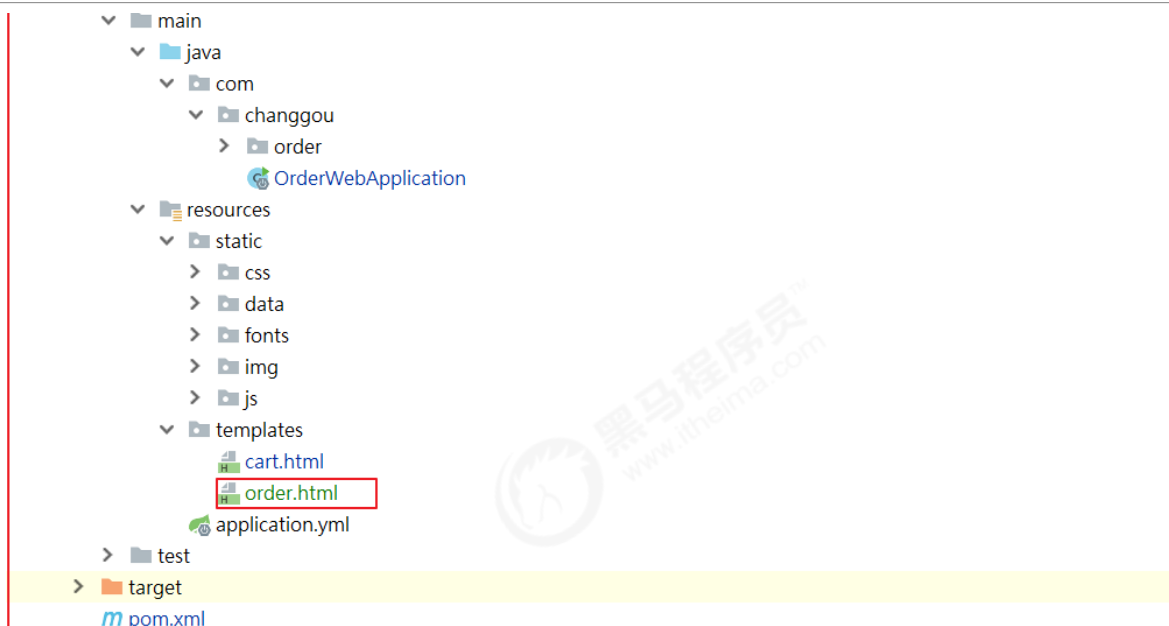


购物车这块也使用的是模板渲染，用户先请求经过微服务网关，微服务网关转发到订单购物车模板渲染服务，模板渲染服务调用用户微服务和订单购物车微服务查询用户收件地址和购物车清单，然后到页面显示。

### 1.3.1 准备工作

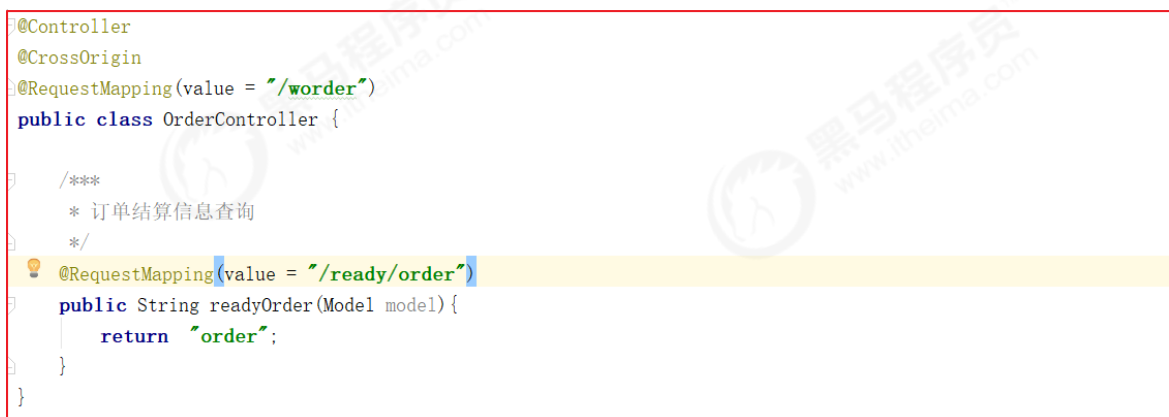
#### (1) 静态资源导入

将资料中的 `order.html` 拷贝到 `changgou-web-order` 工程的 `templates` 中。



## (2)页面跳转实现

在changgou-web-order中创建 `com.changgou.order.controller.OrderController` 实现页面跳转，代码如下：



## (3)网关配置

修改changgou-gateway-web的application.yml文件，将订单的路由过滤地址添加上去，代码如下：



同时不要忘了把该地址添加到登录过滤地址中，修改 `com.changgou.filter.URLFilter`，在 `orderFilterPath`里添加 `/api/worder/**` 过滤，代码如下：





因为一会儿要调用changgou-service-user查询用户的收件地址信息，调用changgou-service-order查询购物车清单信息，所以我们需要创建Feign。购物车的Feign之前已经创建过了，所以只需要创建用户地址相关的即可。

### (1)用户地址信息查询

在changgou-service-user-api中创建AddressFeign，代码如下：

```
@FeignClient(name="user")
@RequestMapping("/address")
public interface AddressFeign {

    /**
     * 查询用户的收件地址信息
     * @return
     */
    @GetMapping(value = "/list")
    Result<List<Address>> list();
}
```

### (2)查询购物车和用户收件地址信息

修改changgou-web-order中的 com.changgou.order.controller.OrderController 的readyOrder方法，在该方法中，使用feign调用查询收件地址信息和用户购物车信息，代码如下：

```
@Controller
@RequestMapping("/worder")
public class OrderController {

    @Autowired
    private AddressFeign addressFeign;

    @Autowired
    private CartFeign cartFeign;

    @GetMapping("/ready/order")
    public String toOrder(Model model){
        List<Address> addressList = addressFeign.list().getData();
        model.addAttribute(s: "address",addressList);

        Map map = cartFeign.list();
        List<OrderItem> orderItemList = (List<OrderItem>) map.get("orderItemList");
        Integer totalMoney = (Integer) map.get("totalMoney");
        Integer totalNum = (Integer) map.get("totalNum");

        model.addAttribute(s: "carts",orderItemList);
        model.addAttribute(s: "totalMoney",totalMoney);
        model.addAttribute(s: "totalNum",totalNum);
        return "order";
    }
}
```

### (3)数据回显

修改order.html，回显收件地址信息和购物车信息，代码如下：

收件地址信息：

```

<div class="con name selected"><a href="javascript:;"><em th:text="${addr.username}"></em><span title="点击取消选择"></span></a></div>
<div class="con address">
    <span class="place" th:text="${addr.address}"></span>
    <span class="phone" th:text="${addr.phone}"></span>
    <span class="base">默认地址</span>
</div>
<div class="clearfix"></div>
</div>
</li>
</ul>
    
```

购物车清单：

```

<div class="sendGoods">
    <span>商品清单：</span>
    <ul class="yui3-g" th:each="cart : ${carts}">
        <li class="yui3-u-1-6">
            <span></span>
        </li>
        <li class="yui3-u-7-12">
            <div class="desc" th:text="${cart.name}"></div>
            <div class="seven">7天无理由退货</div>
        </li>
        <li class="yui3-u-1-12">
            <div class="price">¥<em th:text="${cart.price}"></em></div>
        </li>
        <li class="yui3-u-1-12">
            <div class="num">X<em th:text="${cart.num}"></em></div>
        </li>
        <li class="yui3-u-1-12">
            <div class="exit">有货</div>
        </li>
    </ul>
</div>
    
```

测试效果：

localhost:8001/api/worder/ready/order

填写并核对订单信息

收件人信息

szitheima	深圳中横 13900112222	默认地址
szitheima	深圳庭威 13700221122	默认地址
szitheima	中横大厦 13301212233	默认地址

支付方式


☒ 在线支付
 ☐ 货到付款

送货清单

配送方式：

☐ 天天快递
 配送时间：预计8月10日（周三）09:00-15:00送达

商品清单：

	TCL 50英寸液晶电视 立体声 50英寸 170	¥209	X2	有货
7天无理由退货				

商家留言：

建议留言前请先与商家沟通确认

```

<div class="choose-address" th:each="addr : ${address}"> 默认的收件地址追加selected
<div class="con name" th:classappend="${addr.isDefault}==1?'selected':''"><a href="javascript:;"><em th:text="${addr.contact}"></em><span title="
<div class="con address">
    <span class="place" th:text="${addr.address}"></span>
    <span class="phone" th:text="${addr.phone}"></span>
    <span class="base" th:if="${addr.isDefault}==1">默认地址</span>
</div>
<div class="clearfix"></div>
</div>
    
```

效果如下：

填写并核对订单信息

收件人信息

李慕诚	深圳中粮 13900112222
李佳红	深圳庭威 13700221122
李佳星	中颐大厦 13301212233 <span>默认地址</span>

### 1.3.3 记录选中收件人

用户每次点击收件人的时候，我们需要记录收件人信息。我们可以使用Vue，定义一个订单变量，并且每次点击的时候，将该收件人信息传给Vue的一个方法在订单变量中记录选中的用户信息即可。

(1)引入vue

我们要先引入Vue,在order.html中引入vue，代码如下：

```

<script type="text/javascript" src="/js/vue.js"></script>
<script type="text/javascript" src="/js/axios.js"></script>
    
```

同时在72行左右添加一个id="app"作为Vue入口标签。

```

71
72 <div id="app" class="cart py-container">
73     <!--主内容-->
74     <div class="checkout py-container">
75         ...
76     </div>
77 </div>
    
```

(2)定义记录用户信息方法

```

    el: '#app',
    data: {
      order: {}, //用户提交的订单信息
    },
    methods: {
      //选中的用户联系人、用户收件地址、用户手机号
      chooseaddr: function (contact, mobile, address) {
        app.$set(app.order, 'receiverContact', contact);
        app.$set(app.order, 'receiverMobile', mobile);
        app.$set(app.order, 'receiverAddress', address);
      }
    }
  })
</script>

```

修改地址列表，每次点击的时候调用上面的方法，代码如下：

```

<div class="choose-address" th:each="addr : ${address}">
  <div class="con name" th:click="chooseaddr('${addr.contact}', '${addr.phone}', '${addr.address}')" th:classappend="${addr.isDefault}==1? 'se">
    <div class="con address">
      <span class="place" th:text="${addr.address}"></span>
      <span class="phone" th:text="${addr.phone}"></span>
      <span class="base" th:if="${addr.isDefault}==1">默认地址</span>
    </div>
    <div class="clearfix"></div>
  </div>

```

将选中的地址收件人信息回显到页面输出，代码如下：

```

<div class="clearfix trade">
  <div class="fc-price">应付金额: <span class="price">¥5399.00</span></div>
  <div class="fc-receiverInfo">
    寄送至:
    <span id="receive-address">{{order.receiverAddress}}</span>
    收货人: <span id="receive-name">{{order.receiverContact}}</span>
    <span id="receive-phone">{{order.receiverMobile}}</span>
  </div>
</div>

```

测试效果如下：

应付金额: **¥5399.00**

寄送至: 深圳庭威 收货人: 李佳红 13700221122

提交订单

(3)默认收件人加载

用户没有手动选择收件人信息的时候，收件人信息没有初始化。

应付金额: **¥5399.00**

寄送至: {{order.receiverAddress}} 收货人: {{order.receiverContact}} {{order.receiverMobile}}

提交订单

我们可以在后台加载找出默认的收件人信息，前台通过Vue直接绑定给变量即可。

修改 com.changgou.order.controller.OrderController,添加默认收件人信息判断，代码如下：





```
@RequestMapping(value = "/ready/order")
public String readyOrder(Model model) {
    //查询用户收件地址信息
    Result<List<Address>> address = addressFeign.list();

    //查询用户购物车数据
    Result<List<OrderItem>> carts = cartFeign.list();

    //默认收件人信息
    for (Address addr : address.getData()) {
        if (addr.getIsDefault().equals("1")) {
            model.addAttribute("deAddr", addr);
            break;
        }
    }

    model.addAttribute("address", address.getData());
    model.addAttribute("carts", carts.getData());
    return "order";
}
```

修改order.html，代码如下：

```
<script th:inline="javascript">
    var app = new Vue({
        el: '#app',
        data: {
            //取出默认收件人信息
            order: [{ 'receiverContact': [${deAddr.contact}], 'receiverMobile': [${deAddr.phone}], 'receiverAddress': [${deAddr.address}] }], //用户提交的订单信息
        },
        methods: {
            //选中的用户联系人、用户收件地址、用户手机号
            chooseaddr: function (contact, mobile, address) {
                app.$set(app.order, 'receiverContact', contact);
                app.$set(app.order, 'receiverMobile', mobile);
                app.$set(app.order, 'receiverAddress', address);
            }
        }
    })
</script>
```

此时页面可以正常显示用户信息了。

### 1.3.4 支付方式选中

支付方式为线上支付和货到付款，我们可以在order变量中定义一个属性 payType，点击线上支付让他的值为1，点击货到付款，让他的值为0即可。

定义变量：

```
<script th:inline="javascript">
    var app = new Vue({
        el: '#app',
        data: {
            order: [{ 'receiverContact': [${deAddr.contact}], 'receiverMobile': [${deAddr.phone}], 'receiverAddress': [${deAddr.address}], 'payType': 1 },
        },
        methods: {
            //选中的用户联系人、用户收件地址、用户手机号
            chooseaddr: function (contact, mobile, address) {
                app.$set(app.order, 'receiverContact', contact);
                app.$set(app.order, 'receiverMobile', mobile);
                app.$set(app.order, 'receiverAddress', address);
            }
        }
    })
</script>
```

修改页面，添加点击事件：

```
<li class= selected [th:click= order.payType=1] 在线支付<span title= 点击取消选择 ></span></li>
<li th:click= order.payType=0 >货到付款<span title= 点击取消选择 ></span></li>
</ul>
</div>
```

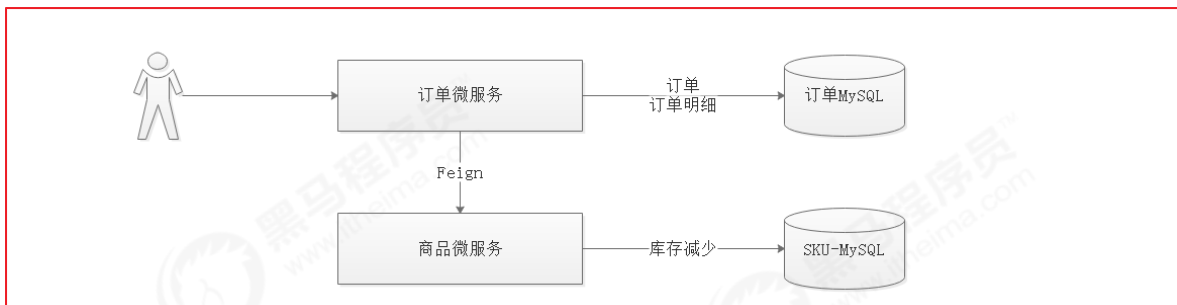
购物车页面对接，修改cart.html页面的结算地址，代码如下：

```
<div class="sumbtn">
  <a class="sum-btn" href="/api/worder/ready/order">结算</a>
</div>
```

## 2 下单

### 2.1 业务分析

点击提交订单的时候，会立即创建订单数据，创建订单数据会将数据存入到2张表中，分别是订单表和订单明细表，此处还需要修改商品对应的库存数量。



订单表结构如下：

```
CREATE TABLE `tb_order` (
  `id` varchar(50) COLLATE utf8_bin NOT NULL COMMENT '订单id',
  `total_num` int(11) DEFAULT NULL COMMENT '数量合计',
  `total_money` int(11) DEFAULT NULL COMMENT '金额合计',
  `pre_money` int(11) DEFAULT NULL COMMENT '优惠金额',
  `post_fee` int(11) DEFAULT NULL COMMENT '邮费',
  `pay_money` int(11) DEFAULT NULL COMMENT '实付金额',
  `pay_type` varchar(1) COLLATE utf8_bin DEFAULT NULL COMMENT '支付类型，1、在线支付、0 货到付款',
  `create_time` datetime DEFAULT NULL COMMENT '订单创建时间',
  `update_time` datetime DEFAULT NULL COMMENT '订单更新时间',
  `pay_time` datetime DEFAULT NULL COMMENT '付款时间',
  `consign_time` datetime DEFAULT NULL COMMENT '发货时间',
  `end_time` datetime DEFAULT NULL COMMENT '交易完成时间',
  `close_time` datetime DEFAULT NULL COMMENT '交易关闭时间',
  `shipping_name` varchar(20) COLLATE utf8_bin DEFAULT NULL COMMENT '物流名称',
  `shipping_code` varchar(20) COLLATE utf8_bin DEFAULT NULL COMMENT '物流单号',
  `username` varchar(50) COLLATE utf8_bin DEFAULT NULL COMMENT '用户名称',
  `buyer_message` varchar(1000) COLLATE utf8_bin DEFAULT NULL COMMENT '买家留言',
  `buyer_rate` char(1) COLLATE utf8_bin DEFAULT NULL COMMENT '是否评价',
  `receiver_contact` varchar(50) COLLATE utf8_bin DEFAULT NULL COMMENT '收货人',
  `receiver_mobile` varchar(12) COLLATE utf8_bin DEFAULT NULL COMMENT '收货人手机',
```

```
`source_type` char(1) COLLATE utf8_bin DEFAULT NULL COMMENT '订单来源: 1:web,
2: app, 3: 微信公众号, 4: 微信小程序 5 H5手机页面',
`transaction_id` varchar(30) COLLATE utf8_bin DEFAULT NULL COMMENT '交易流水号',
`order_status` char(1) COLLATE utf8_bin DEFAULT NULL COMMENT '订单状态,0:未完
成,1:已完成, 2: 已退货',
`pay_status` char(1) COLLATE utf8_bin DEFAULT NULL COMMENT '支付状态,0:未支付,
1: 已支付, 2: 支付失败',
`consign_status` char(1) COLLATE utf8_bin DEFAULT NULL COMMENT '发货状态,0:未发
货, 1: 已发货, 2: 已收货',
`is_delete` char(1) COLLATE utf8_bin DEFAULT NULL COMMENT '是否删除',
PRIMARY KEY (`id`),
KEY `create_time` (`create_time`),
KEY `status` (`order_status`),
KEY `payment_type` (`pay_type`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin;
```

订单明细表结构如下：

```
CREATE TABLE `tb_order_item` (
  `id` varchar(50) COLLATE utf8_bin NOT NULL COMMENT 'ID',
  `category_id1` int(11) DEFAULT NULL COMMENT '1级分类',
  `category_id2` int(11) DEFAULT NULL COMMENT '2级分类',
  `category_id3` int(11) DEFAULT NULL COMMENT '3级分类',
  `spu_id` varchar(20) COLLATE utf8_bin DEFAULT NULL COMMENT 'SPU_ID',
  `sku_id` bigint(20) NOT NULL COMMENT 'SKU_ID',
  `order_id` bigint(20) NOT NULL COMMENT '订单ID',
  `name` varchar(200) COLLATE utf8_bin DEFAULT NULL COMMENT '商品名称',
  `price` int(20) DEFAULT NULL COMMENT '单价',
  `num` int(10) DEFAULT NULL COMMENT '数量',
  `money` int(20) DEFAULT NULL COMMENT '总金额',
  `pay_money` int(11) DEFAULT NULL COMMENT '实付金额',
  `image` varchar(200) COLLATE utf8_bin DEFAULT NULL COMMENT '图片地址',
  `weight` int(11) DEFAULT NULL COMMENT '重量',
  `post_fee` int(11) DEFAULT NULL COMMENT '运费',
  `is_return` char(1) COLLATE utf8_bin DEFAULT NULL COMMENT '是否退货,0:未退货, 1:
已退货',
  PRIMARY KEY (`id`),
  KEY `item_id` (`sku_id`),
  KEY `order_id` (`order_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin;
```

## 2.2 下单实现

下单的时候，先往tb\_order表中增加数据，再往tb\_order\_item表中增加数据。

### 2.2.1 代码实现

这里先修改changgou-service-order微服务，实现下单操作，这里会生成订单号，我们首先需要在启动类中创建一个IdWorker对象。

在 com.changgou.OrderApplication 中创建IdWorker，代码如下：



```
public IdWorker() {  
    return new IdWorker(1,1);  
}
```

### (1)控制层

修改changgou-service-order微服务，修改com.changgou.order.controller.OrderController类，代码如下：

```
@Autowired  
private TokenDecode tokenDecode;  
  
/**  
 * 新增Order数据  
 * @param order  
 * @return  
 */  
@PostMapping  
public Result add(@RequestBody Order order){  
    //获取用户名  
    Map<String, String> userMap = tokenDecode.getUserInfo();  
    String username = userMap.get("username");  
    //设置购买用户  
    order.setUsername(username);  
    orderService.add(order);  
    return new Result(true, StatusCode.OK, "添加成功");  
}
```

### (2)业务层

实现逻辑：

- 1) 获取所有购物项
- 2) 统计计算：总金额，总数量
- 3) 填充订单数据并保存
- 4) 获取每一个购物项保存到orderItem
- 5) 删除购物车中数据

修改订单微服务添加com.changgou.order.service.impl.OrderServiceImpl,代码如下：

```
/**  
 * 增加  
 * @param order  
 */  
@Override  
public void add(Order order){  
    //1) 获取所有购物项  
    Map cartMap = cartService.list(order.getUsername());  
    List<OrderItem> orderItemList = (List<OrderItem>)  
cartMap.get("orderItemList");  
    //3) 填充订单数据并保存  
    order.setTotalNum((Integer) cartMap.get("totalNum"));  
    order.setTotalMoney((Integer) cartMap.get("totalMoney"));  
}
```

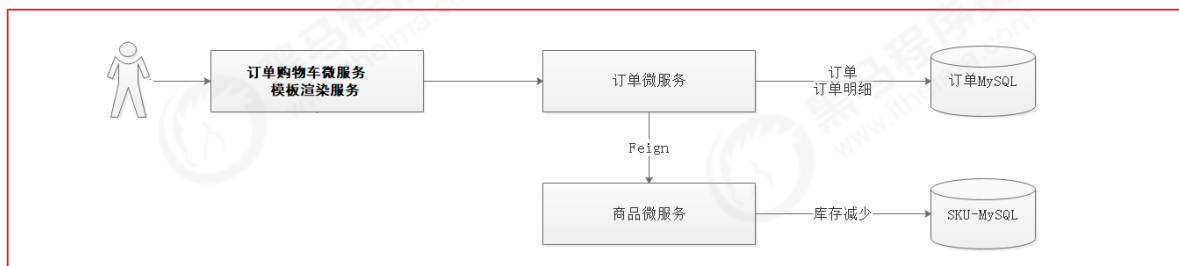
```

        order.setUpdateTime(order.getCreateTime());
        order.setBuyerRate("0");           //0:未评价, 1: 已评价
        order.setSourceType("1");           //来源, 1: WEB
        order.setOrderStatus("0");          //0:未完成, 1:已完成, 2: 已退货
        order.setPayStatus("0");            //0:未支付, 1: 已支付, 2: 支付失败
        order.setConsignStatus("0");        //0:未发货, 1: 已发货, 2: 已收货
        order.setId(idWorker.nextId()+"");
        int count = orderMapper.insertSelective(order);

        //添加订单明细
        for (OrderItem orderItem : orderItemList) {
            orderItem.setId(idWorker.nextId()+"");
            orderItem.setIsReturn("0");
            orderItem.setOrderId(order.getId());
            orderItemMapper.insertSelective(orderItem);
        }

        //清除Redis缓存购物车数据
        redisTemplate.delete("Cart_"+order.getUsername());
    }
    
```

## 2.2.2 渲染服务对接



我们需要在模板渲染端调用订单微服务实现下单操作,下单操作需要调用订单微服务,所以需要创建对应的Feign。

### (1)Feign创建

修改changgou-service-order-api, 添加OrderFeign, 代码如下:

```

@FeignClient(name="order")
public interface OrderFeign {

    /**
     * 提交订单数据
     * @param order
     * @return
     */
    @PostMapping("/order")
    Result add(@RequestBody Order order);
}
    
```

### (2)下单调用



```
@Autowired
private OrderFeign orderFeign;

/**
 * 添加订单数据到购物车中
 * @param order
 * @return
 */
@PostMapping(value = "/add")
@ResponseBody
public Result add(@RequestBody Order order){
    Result result = orderFeign.add(order);
    return result;
}
```

### (3)页面调用

修改order.html，增加下单js方法，并且在页面点击下单调用，代码如下：

```
var app = new Vue({
  el: '#app',
  data: {
    order: { 'receiverContact': [`${deAddr.contact}`], 'receiverMobile': [`${deAddr.phone}`], 'receiverAddress': [`${deAddr.address}`], 'payType': 1 },
  },
  methods: {
    //选中的用户联系人、用户收件地址、用户手机号
    chooseaddr: function (contact, mobile, address) {
      app.$set(app.order, 'receiverContact', contact);
      app.$set(app.order, 'receiverMobile', mobile);
      app.$set(app.order, 'receiverAddress', address);
    },
    //保存订单数据
    add: function () {
      axios.post('/api/worder/add', this.order).then(function (response) {
        if (response.data.flag) {
          //添加成功
          alert('添加成功')
        } else {
          alert('添加失败')
        }
      })
    }
  }
})
```

点击提交订单调用

```
<div class="submit">
  <a class="sui-btn btn-danger btn-xlarge" href="javascript:void(0)" @click="add()">提交订单</a>
</div>
```

保存订单测试，表数据变化如下：

tb\_order表数据：

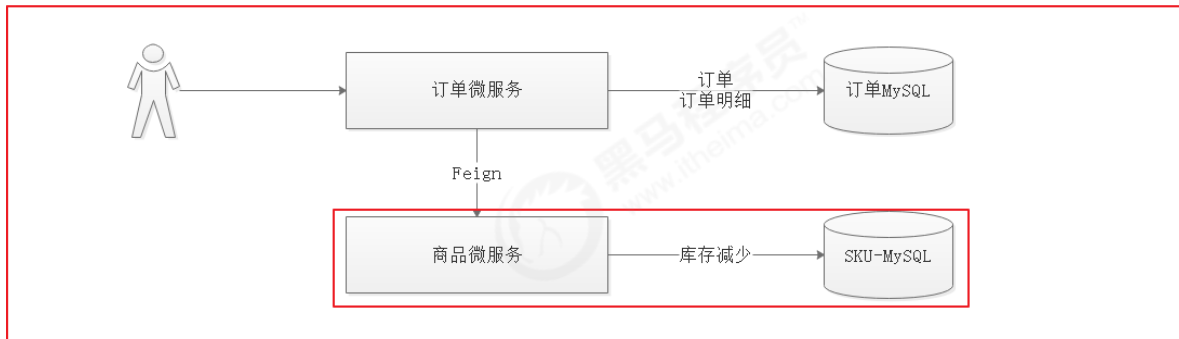
对象	tb_order_item @changgou...	tb_order @changgou_order...	无标题 @changgou_goods...
	开始事务	备注	筛选
	排序	导入	导出
id	total_num	total_money	pre_money
NO.113023765228985139 2		1500000	50
			(Null)
		1499950	1
		2019-05-19 22:23:50	2019-05-19 22:23:50

tb\_order\_item表数据：

对象	tb_order_item @changgou...	tb_order @changgou_order...	无标题 @changgou_goods...
	开始事务	备注	筛选
	排序	导入	导出
sku_id	order_id	name	price
1087918019495202816	NO.113023765228985139	这个是商品名称红64G	900000
1087964956357431296	NO.113023765228985139	xxxxxxx 绿 8G	600000
			1
			600000
			599980
			http://img10.360buyimg.c

### 2.3.1 业务分析

上面操作只实现了下单操作，但对应的库存还没跟着一起减少，我们在下单之后，应该调用商品微服务，将下单的商品库存减少，销量增加。每次订单微服务只需要将用户名传到商品微服务，商品微服务通过用户名到Redis中查询对应的购物车数据，然后执行库存减少，库存减少需要控制当前商品库存 $\geq$ 销售数量。



如何控制库存数量 $\geq$ 购买数量呢？其实可以通过SQL语句实现，每次减少数量之前，加个条件判断。

`where num $\geq$ {num}` 即可。

商品服务需要查询购物车数据，所以需要引入订单的api，在pom.xml中添加如下依赖：

```
<!--order api 依赖-->
<dependency>
  <groupId>com.changgou</groupId>
  <artifactId>changgou_service_order_api</artifactId>
  <version>1.0-SNAPSHOT</version>
</dependency>
```

### 2.3.2 代码实现

要调用其他微服务，需要将头文件中的令牌数据携带到其他微服务中取，所以我们不能使用hystrix的多线程模式，修改changgou-service-order的applicatin.yml配置，代码如下：

```
#hystrix 配置
hystrix:
  command:
    default:
      execution:
        isolation:
          thread:
            timeoutInMilliseconds: 10000
          strategy: SEMAPHORE
```

每次还需要使用拦截器添加头文件信息，修改配置类com.changgou.OrderApplication添加拦截器，代码如下：

```
@Bean
public FeignInterceptor feignInterceptor(){
    return new FeignInterceptor();
}
```





```
/**
 * 递减库存
 * @param orderItem
 * @return
 */
@update("UPDATE tb_sku SET num=num-#{num},sale_num=sale_num+#{num} WHERE id=#{skuId} AND num>=#{num}")
int decrCount(OrderItem orderItem);
```

## (2)业务层

修改changgou-service-order微服务的 com.changgou.goods.service.SkuService 接口，添加如下方法：

```
/**
 * 库存递减
 * @param username
 */
void decrCount(String username);
```

修改changgou-service-order微服务的 com.changgou.goods.service.impl.SkuServiceImpl 实现类，添加一个实现方法，代码如下：

```
@Autowired
private RedisTemplate redisTemplate;

/**
 * 库存递减
 * @param username
 */
@Override
public void decrCount(String username) {
    //获取购物车数据
    List<OrderItem> orderItems = redisTemplate.boundHashOps("Cart_" + username).values();

    //循环递减
    for (OrderItem orderItem : orderItems) {
        //递减库存
        int count = skuMapper.decrCount(orderItem);
        if(count<=0){
            throw new RuntimeException("库存不足，递减失败！");
        }
    }
}
```

## (3)控制层

修改changgou-service-goods的 com.changgou.goods.controller.SkuController 类，添加库存递减方法，代码如下：





```
/*
 * @param username
 * @return
 */
@PostMapping(value = "/decr/count")
public Result decrCount(@RequestParam("username") String username){
    //库存递减
    skuService.decrCount(username);
    return new Result(true, StatusCode.OK, "库存递减成功!");
}
```

#### (4)创建feign

同时在changgou-service-goods-api工程添加 com.changgou.goods.feign.SkuFeign 的实现，代码如下：

```
/**
 * 库存递减
 * @param username
 * @return
 */
@PostMapping(value = "/decr/count")
Result decrCount(@RequestParam(value = "username") String username);
```

### 2.3.3 调用库存递减

修改changgou-service-order微服务的com.changgou.order.service.impl.OrderServiceImpl类的add方法，增加库存递减的调用。

先注入SkuFeign

```
@Autowired
private SkuFeign skuFeign;
```

再调用库存递减方法

```
//库存减库存
skuFeign.decrCount(order.getUsername());
```

完整代码如下：

```

*/
@Override
public void add(Order order) {
    //查询出用户的所有购物车
    //..略

    //添加订单明细
    for (OrderItem orderItem : orderItems) {
        orderItem.setId("NO."+idWorker.nextId());
        orderItem.setIsReturn("0");
        orderItem.setOrderId(order.getId());
        orderItemMapper.insertSelective(orderItem);
    }

    //库存减库存
    skuFeign.decrCount(order.getUsername());

    //清除Redis缓存购物车数据
    redisTemplate.delete(key: "Cart_"+order.getUsername());
}

```

### 2.3.4 测试

库存减少前，查询数据库Sku数据如下：个数98，销量0

```

1 SELECT id,num,sale_num FROM tb_sku WHERE id IN('1087918019495202816','1087964956357431296');
2
3

```

id	num	sale_num
1087918019495202816	98	0
1087964956357431296	98	0

使用Postman执行 <http://localhost:18081/api/order/add>

POST http://localhost:18081/api/order/add

Body: raw, JSON (application/json)

```

1 {
2   "payType": "1",
3   "receiverContact": "张三",
4   "receiverMobile": "13670000000"
5 }

```

Status: 200 OK Time: 61 ms Size: 378 B

Body: Pretty, Raw, Preview, JSON

```

1 {
2   "flag": true,
3   "code": 200000,
4   "message": "下单成功!",
5   "data": null
6 }

```

执行测试后，剩余库存97，销量1

```

1 SELECT id,num,sale_num FROM tb_sku WHERE id IN('1087918019495202816','1087964956357431296');
2
3

```

id	num	sale_num
1087918019495202816	97	1
1087964956357431296	97	1

## 2.4 增加积分(学员练习)



```
CREATE TABLE `tb_user` (  
  `username` varchar(50) NOT NULL COMMENT '用户名',  
  `password` varchar(100) NOT NULL COMMENT '密码，加密存储',  
  `phone` varchar(20) DEFAULT NULL COMMENT '注册手机号',  
  `email` varchar(50) DEFAULT NULL COMMENT '注册邮箱',  
  `created` datetime NOT NULL COMMENT '创建时间',  
  `updated` datetime NOT NULL COMMENT '修改时间',  
  `source_type` varchar(1) DEFAULT NULL COMMENT '会员来源: 1:PC, 2: H5, 3:  
Android, 4: IOS',  
  `nick_name` varchar(50) DEFAULT NULL COMMENT '昵称',  
  `name` varchar(50) DEFAULT NULL COMMENT '真实姓名',  
  `status` varchar(1) DEFAULT NULL COMMENT '使用状态 (1正常 0非正常)',  
  `head_pic` varchar(150) DEFAULT NULL COMMENT '头像地址',  
  `qq` varchar(20) DEFAULT NULL COMMENT 'QQ号码',  
  `is_mobile_check` varchar(1) DEFAULT '0' COMMENT '手机是否验证 (0否 1是)',  
  `is_email_check` varchar(1) DEFAULT '0' COMMENT '邮箱是否检测 (0否 1是)',  
  `sex` varchar(1) DEFAULT '1' COMMENT '性别, 1男, 0女',  
  `user_level` int(11) DEFAULT NULL COMMENT '会员等级',  
  `points` int(11) DEFAULT NULL COMMENT '积分',  
  `experience_value` int(11) DEFAULT NULL COMMENT '经验值',  
  `birthday` datetime DEFAULT NULL COMMENT '出生年月日',  
  `last_login_time` datetime DEFAULT NULL COMMENT '最后登录时间',  
  PRIMARY KEY (`username`),  
  UNIQUE KEY `username` (`username`) USING BTREE  
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='用户表';
```

### 3.4.1 代码实现

#### (1)dao层

修改changgou-service-user微服务的com.changgou.user.dao.UserMapper接口，增加用户积分方法，代码如下：

```
/**  
 * 增加用户积分  
 * @param username  
 * @param pint  
 * @return  
 */  
@Update("UPDATE tb_user SET points=points+#{point} WHERE username=#{username}")  
int addUserPoints(@Param("username") String username, @Param("point") Integer  
pint);
```

#### (2)业务层

修改changgou-service-user微服务的com.changgou.user.service.UserService接口，代码如下：



```
1999/12/17 16:22
* @param username
* @param pint
* @return
*/
int addUserPoints(String username,Integer pint);
```

修改changgou-service-user微服务的 `com.changgou.user.service.impl.UserServiceImpl`，增加添加积分方法实现，代码如下：

```
/**
 * 修改用户积分
 * @param username
 * @param pint
 * @return
 */
@Override
public int addUserPoints(String username, Integer pint) {
    return userMapper.addUserPoints(username,pint);
}
```

### (3)控制层

修改changgou-service-user微服务的 `com.changgou.user.controller.UserController`，添加增加用户积分方法，代码如下：

```
@Autowired
private TokenDecode tokenDecode;

/**
 * 增加用户积分
 * @param points:要添加的积分
 */
@GetMapping(value = "/points/add")
public Result addPoints(Integer points){
    //获取用户名
    Map<String, String> userMap = tokenDecode.getUserInfo();
    String username = userMap.get("username");

    //添加积分
    userService.addUserPoints(username,points);
    return new Result(true,StatusCode.OK,"添加积分成功!");
}
```

### (4)Feign添加

修改changgou-service-user-api工程，修改 `com.changgou.user.feign.UserFeign`，添加增加用户积分方法，代码如下：

```
1992/12/17 14:11
* @param points
* @return
*/
@GetMapping(value = "/points/add")
Result addPoints(@RequestParam(value = "points")Integer points);
```

### 3.4.2 增加积分调用

修改changgou-service-order，添加changgou-service-user-api的依赖，修改pom.xml,添加如下依赖：

```
<!--user api 依赖-->
<dependency>
    <groupId>com.changgou</groupId>
    <artifactId>changgou-service-user-api</artifactId>
    <version>1.0-SNAPSHOT</version>
</dependency>
```

在增加订单的时候，同时添加用户积分，修改changgou-service-order微服务的com.changgou.order.service.impl.OrderServiceImpl下单方法，增加调用添加积分方法，代码如下：

```
@Autowired
private UserFeign userFeign;

/**
 * 增加Order
 * @param order
 */
@Override
public void add(Order order) {
    //查询出用户的所有购物车
    //.. 略

    //库存减库存
    skuFeign.decrCount(order.getUsername());

    //增加用户积分
    userFeign.addPoints(10);

    //清除Redis缓存购物车数据
    redisTemplate.delete(key: "Cart_" + order.getUsername());
}
```

修改changgou-service-order的启动类 com.changgou.OrderApplication，添加feign的包路径：

```
@SpringBootApplication
@EnableFeignClients(basePackages = {"com.changgou.user.feign", "com.changgou.goods.feign"})
@ComponentScan(basePackages = {"com.changgou.order.dao"})
public class OrderApplication {
```