# 今日大纲

1. 实现购物车
2. 基于Mysql实现读写分离

# 购物车

## 需求描述

1. 用户可以在登录状态下将商品添加到购物车
2. 用户可以在未登录状态下将商品添加到购物车
3. 用户可以使用购物车一起结算下单
4. 用户可以查询自己的购物车
5. 用户可以在购物车中可以修改购买商品的数量。
6. 用户可以在购物车中删除商品。

开发模式：敏捷开发

2个核心：

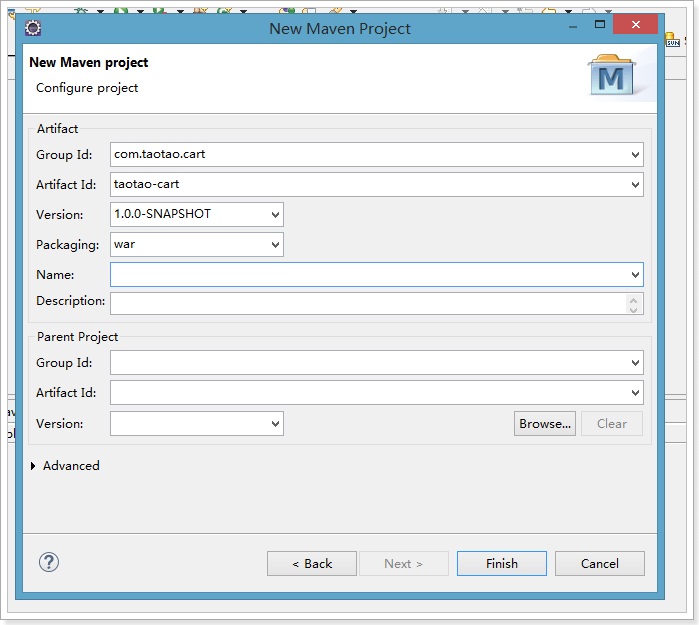
1. 用户故事
2. 周期迭代

## 业务流程



## 搭建购物车系统（taotao-cart）

### 创建工程



### 导入依赖

<project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>com.taotao.parent</groupId>

<artifactId>taotao-parent</artifactId>

<version>0.0.1-SNAPSHOT</version>

</parent>

<groupId>com.taotao.cart</groupId>

<artifactId>taotao-cart</artifactId>

<version>1.0.0-SNAPSHOT</version>

<packaging>war</packaging>

<dependencies>

<dependency>

<groupId>com.taotao.common</groupId>

<artifactId>taotao-common</artifactId>

<version>1.0.0-SNAPSHOT</version>

</dependency>

<!-- 单元测试 -->

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aspects</artifactId>

</dependency>

<!-- Mybatis -->

<dependency>

<groupId>org.mybatis</groupId>

<artifactId>mybatis</artifactId>

</dependency>

<dependency>

<groupId>org.mybatis</groupId>

<artifactId>mybatis-spring</artifactId>

</dependency>

<!-- 分页助手 -->

<dependency>

<groupId>com.github.pagehelper</groupId>

<artifactId>pagehelper</artifactId>

</dependency>

<dependency>

<groupId>com.github.jsqlparser</groupId>

<artifactId>jsqlparser</artifactId>

</dependency>

<!-- 通用Mapper -->

<dependency>

<groupId>com.github.abel533</groupId>

<artifactId>mapper</artifactId>

</dependency>

<!-- MySql -->

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-log4j12</artifactId>

</dependency>

<!-- Jackson Json处理工具包 -->

<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-databind</artifactId>

</dependency>

<!-- 连接池 -->

<dependency>

<groupId>com.jolbox</groupId>

<artifactId>bonecp-spring</artifactId>

<version>0.8.0.RELEASE</version>

</dependency>

<!-- httpclient -->

<dependency>

<groupId>org.apache.httpcomponents</groupId>

<artifactId>httpclient</artifactId>

</dependency>

<!-- JSP相关 -->

<dependency>

<groupId>jstl</groupId>

<artifactId>jstl</artifactId>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>servlet-api</artifactId>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jsp-api</artifactId>

<scope>provided</scope>

</dependency>

<!-- Apache工具组件 -->

<dependency>

<groupId>org.apache.commons</groupId>

<artifactId>commons-lang3</artifactId>

</dependency>

<dependency>

<groupId>org.apache.commons</groupId>

<artifactId>commons-io</artifactId>

</dependency>

</dependencies>

<build>

<plugins>

<!-- 配置Tomcat插件 -->

<plugin>

<groupId>org.apache.tomcat.maven</groupId>

<artifactId>tomcat7-maven-plugin</artifactId>

<configuration>

<port>8086</port>

<path>/</path>

</configuration>

</plugin>

</plugins>

</build>

</project>

### Web.xml

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns=*"http://java.sun.com/xml/ns/javaee"*

xsi:schemaLocation=*"http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"*

id=*"WebApp\_ID"* version=*"2.5"*>

<display-name>taotao-cart</display-name>

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath:spring/applicationContext\*.xml</param-value>

</context-param>

<!--Spring的ApplicationContext 载入 -->

<listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

</listener>

<!-- 编码过滤器，以UTF8编码 -->

<filter>

<filter-name>encodingFilter</filter-name>

<filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>

<init-param>

<param-name>encoding</param-name>

<param-value>UTF8</param-value>

</init-param>

</filter>

<filter-mapping>

<filter-name>encodingFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

<!-- 配置SpringMVC框架入口 -->

<servlet>

<servlet-name>taotao-cart</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<init-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath:spring/taotao-cart-servlet.xml</param-value>

</init-param>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>taotao-cart</servlet-name>

<url-pattern>\*.html</url-pattern>

</servlet-mapping>

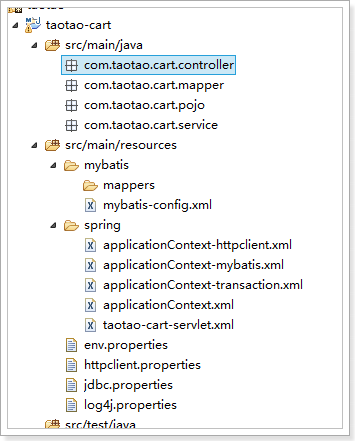
<welcome-file-list>

<welcome-file>index.jsp</welcome-file>

</welcome-file-list>

</web-app>

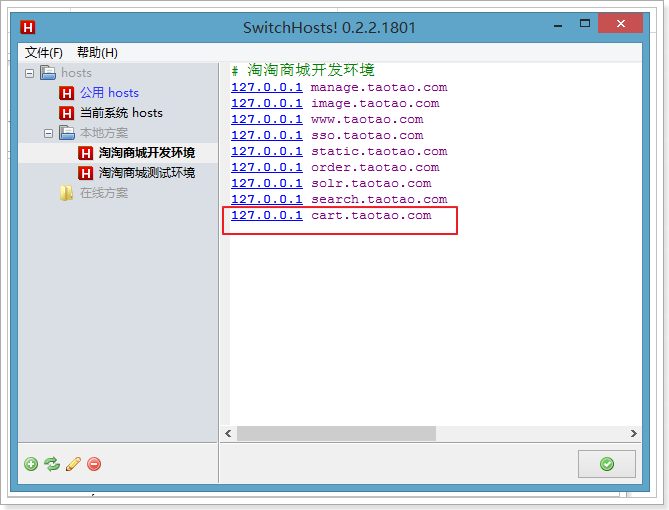
### 其他配置文件



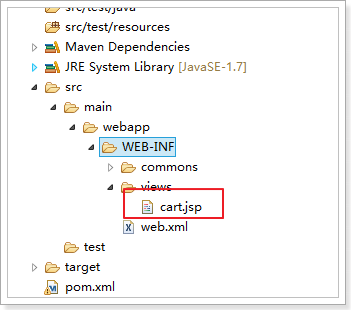
## 配置nginx和hosts



Hosts：



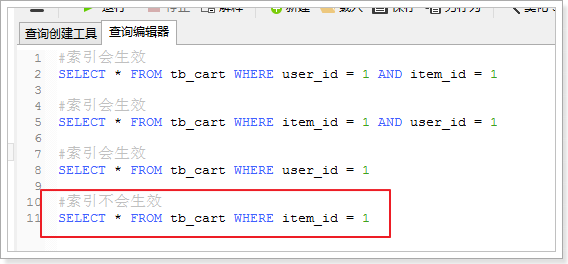
### 导入页面



## 表结构



使用联合索引，一定要注意索引字段的顺序。



## 加入商品到购物车的地址

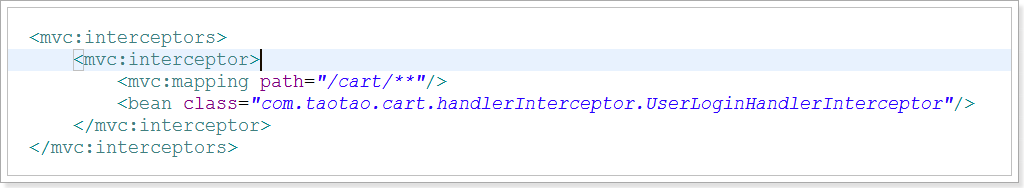
[http://cart.taotao.com/cart/{itemId}.html](http://cart.taotao.com/cart/%7bitemId%7d.html)

## 通过拦截器判断用户是否登录

### 编写拦截器



### 配置拦截器



## 商品加入购物车

### Controller



### Service

加入商品到购物车：

**public** **void** addItemToCart(Long itemId) {

// 判断该商品在购物车中是否存在

User user = UserThreadLocal.*get*();

Cart record = **new** Cart();

record.setItemId(itemId);

record.setUserId(user.getId());

Cart cart = **this**.cartMapper.selectOne(record);

**if** (**null** == cart) {

// 购物车中不存在该商品

cart = **new** Cart();

cart.setItemId(itemId);

cart.setUserId(user.getId());

cart.setNum(1); // **TODO** 先默认为1

cart.setCreated(**new** Date());

cart.setUpdated(cart.getCreated());

Item item = **this**.itemService.queryItemById(itemId);

cart.setItemImage(item.getImages()[0]);

cart.setItemPrice(item.getPrice());

cart.setItemTitle(item.getTitle());

// 将Cart保存到数据库

**this**.cartMapper.insert(cart);

} **else** {

// 该商品已经存在购物车中

cart.setNum(cart.getNum() + 1); // **TODO** 先默认为1

cart.setUpdated(**new** Date());

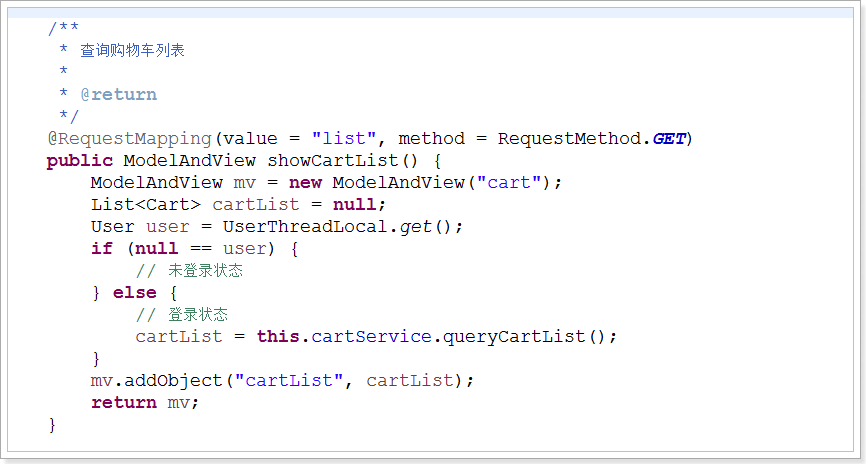
**this**.cartMapper.updateByPrimaryKey(cart);

}

}

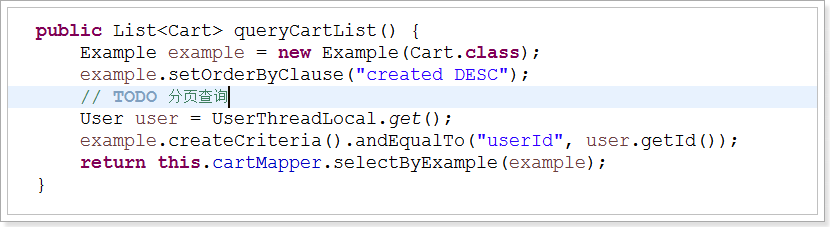
## 查询购物车列表

### Controller



### Service

按照加入购物车时间倒序排序。

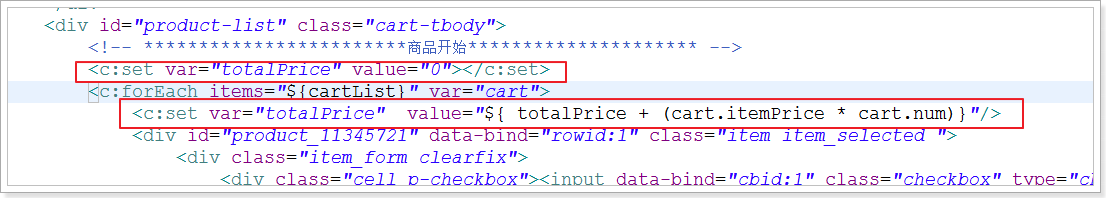


## 测试

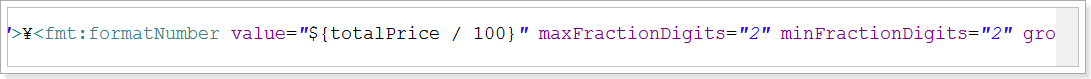


## 计算总价

计算总价：



显示总价：



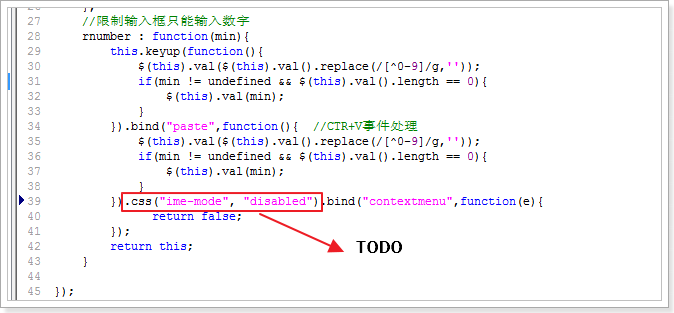
效果：



# 修改购买数量

## JS

### 限制用户输入



解释：<http://www.cnblogs.com/PeunZhang/archive/2012/08/16/2642084.html>

使用：

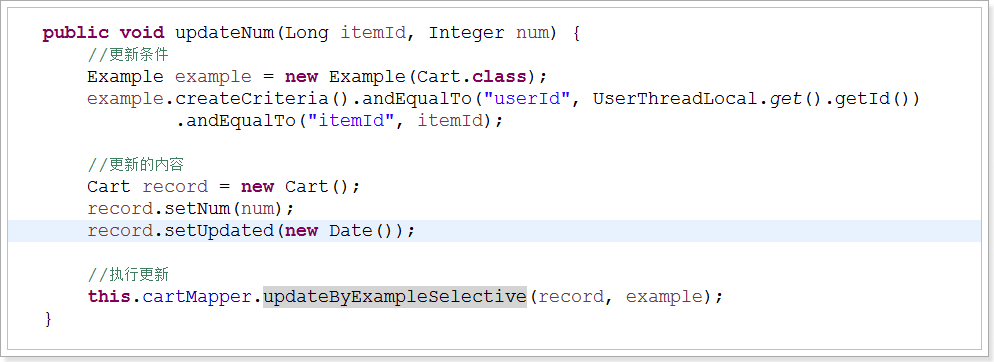


## 后台实现

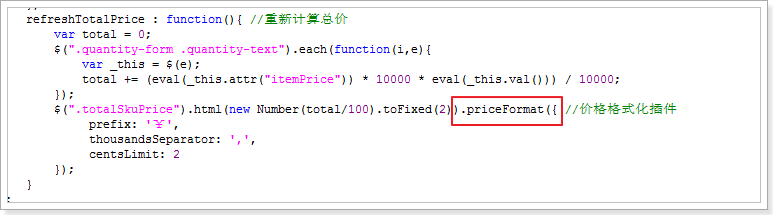
### Controller

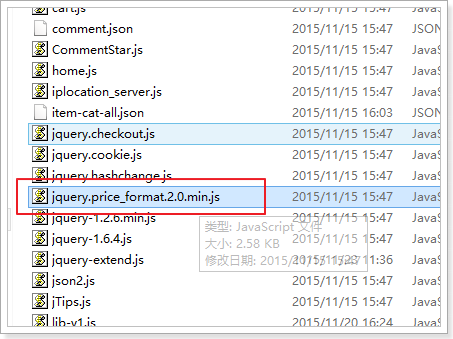


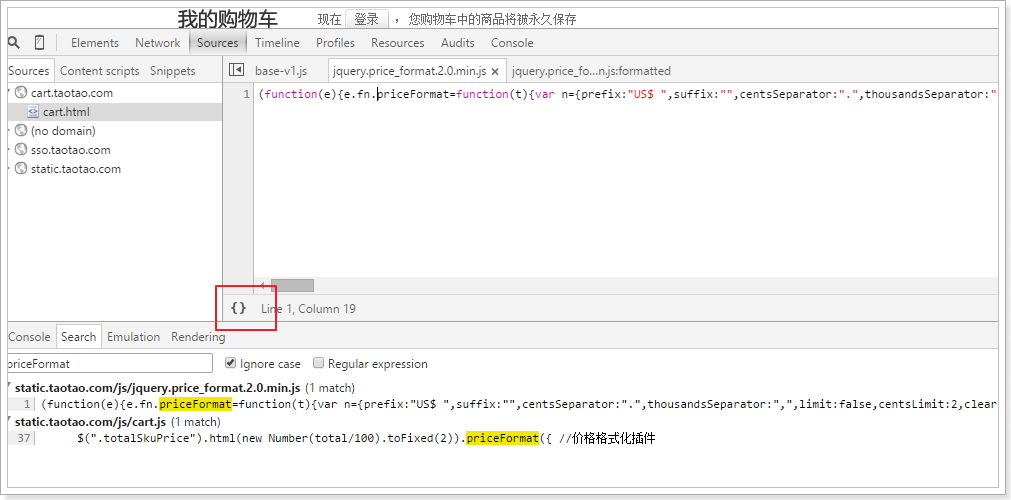
### Service

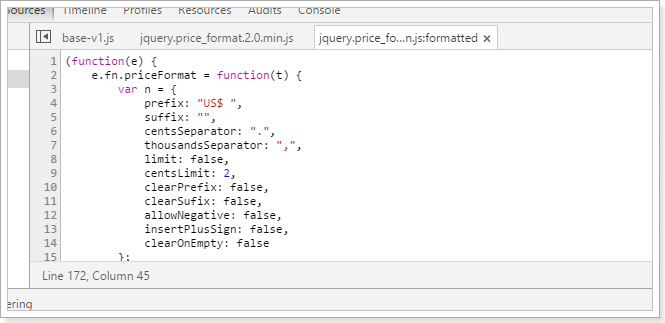


### 格式化价格





格式化压缩的js：  


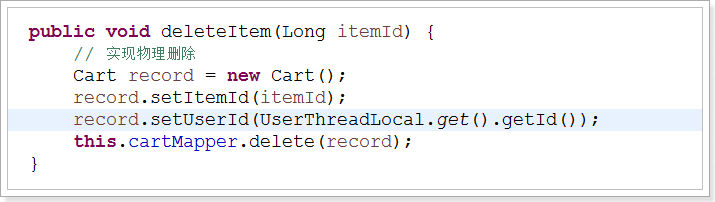


# 删除购物车中商品

## Controller



## Service



# 未登录状态下的购物车

## 以什么样数据格式保存购物车数据到cookie？

可以使用json数据格式。

[

{

itemId:1001

itemTitle:”小米手机手机中的战斗机，欧耶”

……

}

]

## Service实现

**package** com.taotao.cart.service;

**import** java.util.ArrayList;

**import** java.util.Date;

**import** java.util.List;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

**import** org.apache.commons.lang3.StringUtils;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Service;

**import** com.fasterxml.jackson.databind.ObjectMapper;

**import** com.taotao.cart.bean.Item;

**import** com.taotao.cart.pojo.Cart;

**import** com.taotao.common.util.CookieUtils;

@Service

**public** **class** CartCookieService {

**public** **static** **final** String ***COOKIE\_NAME*** = "TT\_CART";

**public** **static** **final** Integer ***COOKIE\_TIME*** = 60 \* 60 \* 24 \* 30 \* 12;

**private** **static** **final** ObjectMapper ***MAPPER*** = **new** ObjectMapper();

@Autowired

**private** ItemService itemService;

/\*\*

\* 查询商品列表, **TODO**: 按照创建时间倒序排序

\*

\* **@param** request

\* **@return**

\*/

**public** List<Cart> queryCartList(HttpServletRequest request) {

String cookieValue = CookieUtils.*getCookieValue*(request, ***COOKIE\_NAME***, **true**);

**if** (StringUtils.*isEmpty*(cookieValue)) {

**return** **new** ArrayList<Cart>(0);

}

**try** {

**return** ***MAPPER***.readValue(cookieValue,

***MAPPER***.getTypeFactory().constructCollectionType(List.**class**, Cart.**class**));

} **catch** (Exception e) {

e.printStackTrace();

}

**return** **new** ArrayList<Cart>(0);

}

**public** **void** addItemToCart(Long itemId, HttpServletRequest request, HttpServletResponse response) {

// 判断该商品在购物车中是否存在，如果存在数量相加，不存在，直接添加

List<Cart> carts = **this**.queryCartList(request);

Cart cart = **null**;

**for** (Cart c : carts) {

**if** (c.getItemId().longValue() == itemId.longValue()) {

cart = c;

}

}

**if** (cart == **null**) {

// 不存在

cart = **new** Cart();

cart.setCreated(**new** Date());

cart.setUpdated(cart.getCreated());

cart.setItemId(itemId);

cart.setNum(1); // **TODO** 默认为1

Item item = **this**.itemService.queryItemById(itemId);

cart.setItemTitle(item.getTitle());

cart.setItemPrice(item.getPrice());

cart.setItemImage(item.getImages()[0]);

carts.add(cart);

} **else** {

// 存在

cart.setNum(cart.getNum() + 1); // **TODO** 默认为1

cart.setUpdated(**new** Date());

}

saveCartsToCookie(request, response, carts);

}

**private** **void** saveCartsToCookie(HttpServletRequest request, HttpServletResponse response, List<Cart> carts) {

**try** {

// 将购物车数据写入到cookie中

CookieUtils.*setCookie*(request, response, ***COOKIE\_NAME***, ***MAPPER***.writeValueAsString(carts),

***COOKIE\_TIME***, **true**);

} **catch** (Exception e) {

e.printStackTrace();

}

}

**public** **void** udpateNum(Long itemId, Integer num, HttpServletRequest request, HttpServletResponse response) {

List<Cart> carts = **this**.queryCartList(request);

**for** (Cart cart : carts) {

**if** (cart.getItemId().longValue() == itemId.longValue()) {

cart.setNum(num);

cart.setUpdated(**new** Date());

**break**;

}

}

saveCartsToCookie(request, response, carts);

}

**public** **void** deleteItem(Long itemId, HttpServletRequest request, HttpServletResponse response) {

List<Cart> carts = **this**.queryCartList(request);

**for** (Cart cart : carts) {

**if** (cart.getItemId().longValue() == itemId.longValue()) {

carts.remove(cart);

**break**;

}

}

saveCartsToCookie(request, response, carts);

}

}

# 基于购物车实现下单功能

## 购物车页面中跳转到前端的订单确认页



## 前端系统的中实现



通过购物车系统提供的接口查询数据：



购物车系统中开发接口，根据用户id查询购物车列表：



订单确认页：



关键点：form表单



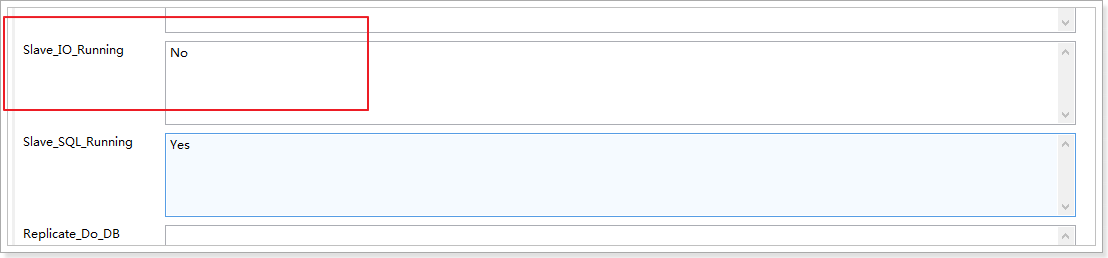
下单成功:



# 读写分离

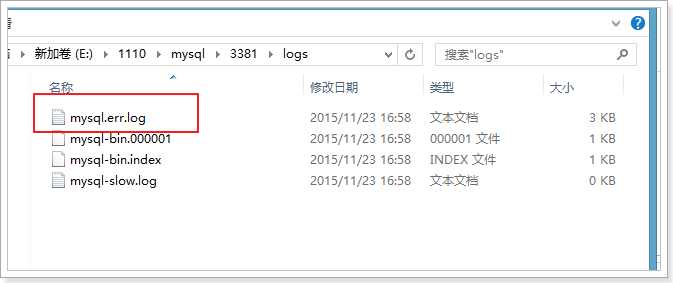
《使用Spring实现读写分离（MySQL实现主从复制）.docx》

## 主从设置失败

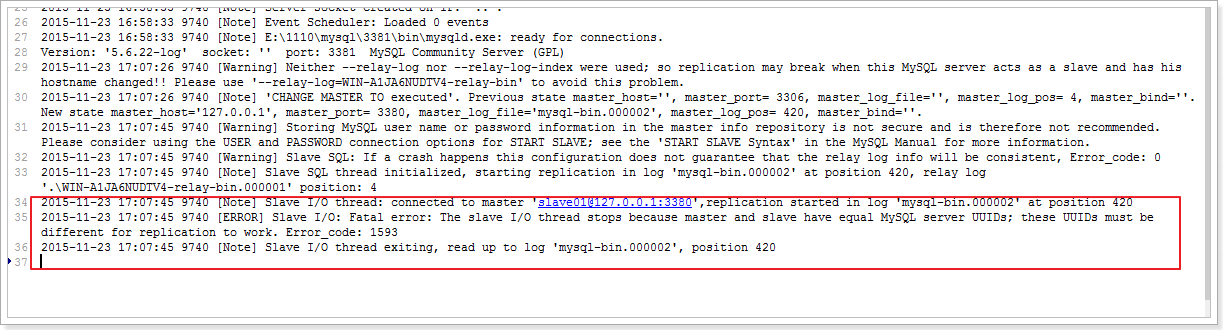


如何定位问题？

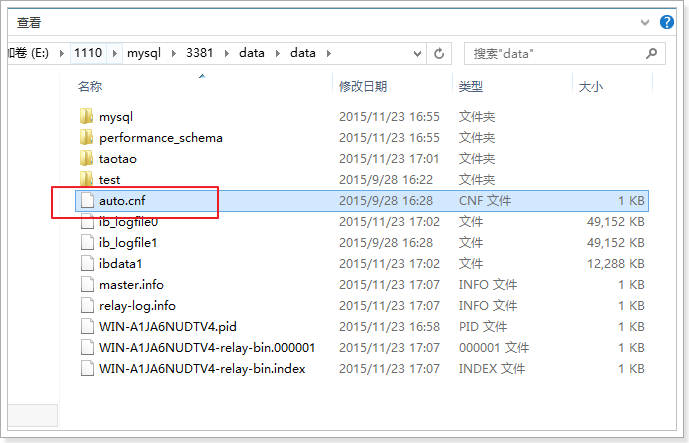
查看3381的日志：

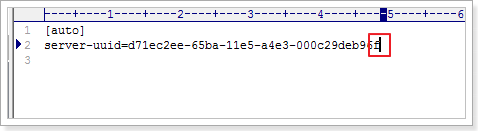


问题：



如何设置Mysql的UUID？





问题解决：

