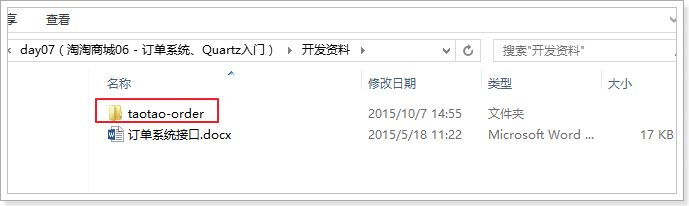
# 今日大纲

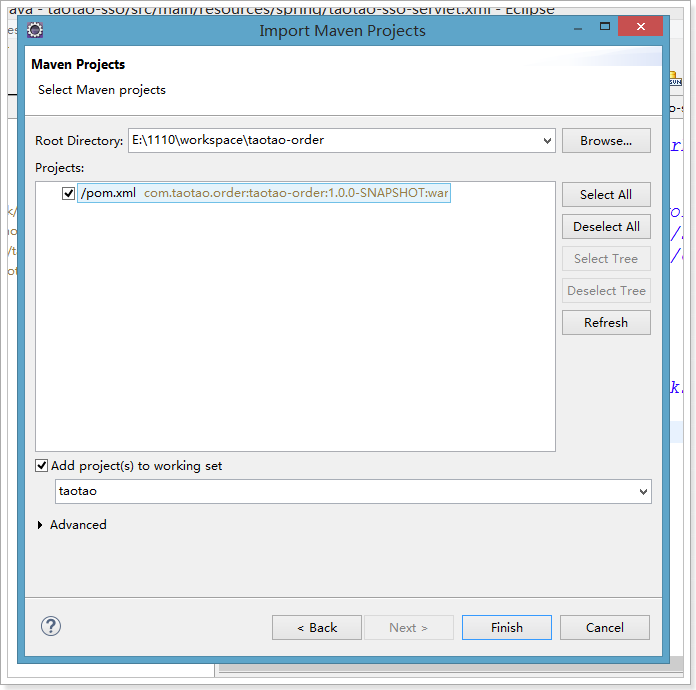
1. 讲解订单系统
2. 基于订单系统完成下单功能的开发
3. 使用Solr完成商品的搜索功能

# 订单系统

说明：订单系统只是做讲解，不做开发。

## 导入taotao-order



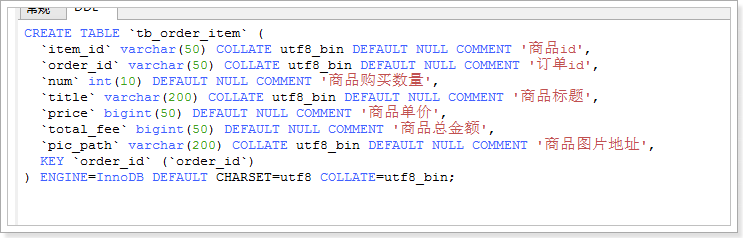


## 表结构

订单表：



订单商品表：



疑问：为什么要冗余存储商品的基本信息？

订单物流表（收货人信息）：



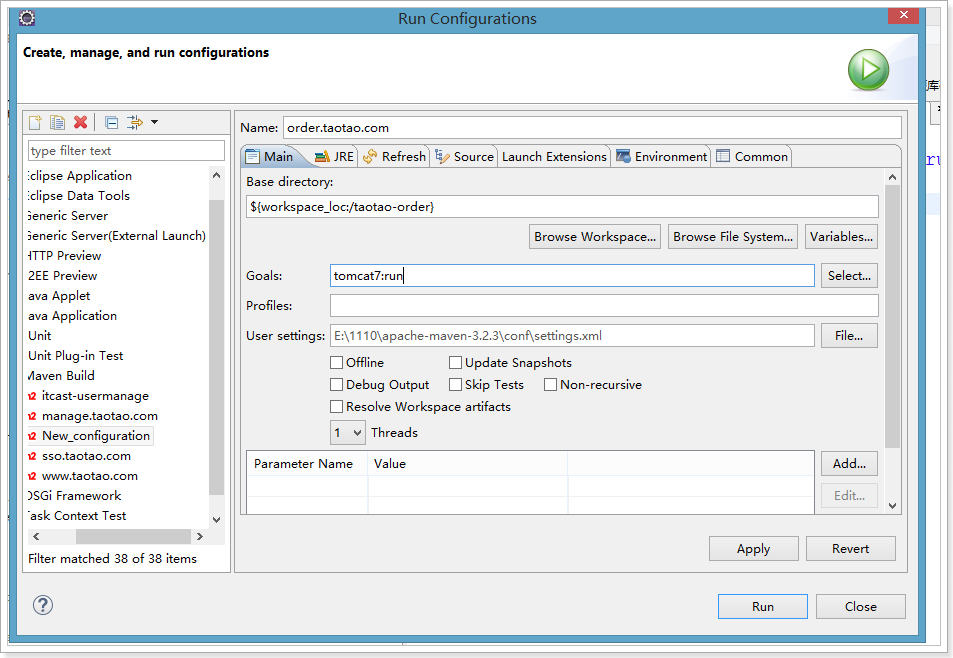
## 开放平台



<http://open.taobao.com/doc2/apiDetail?spm=0.0.0.0.MiNiKt&apiId=47&docType>=：



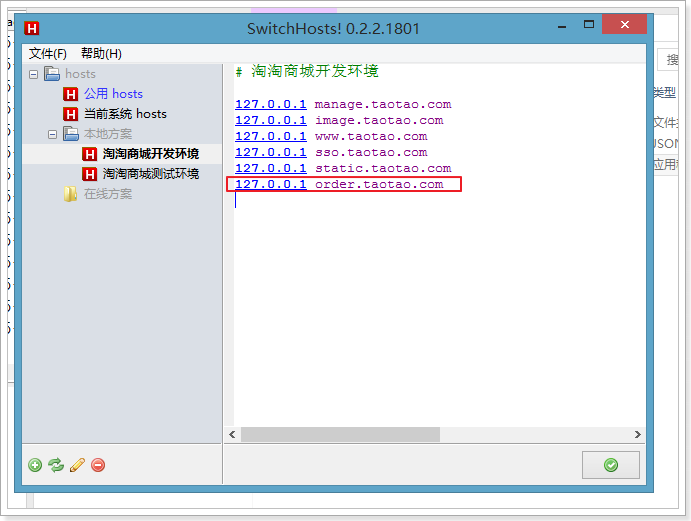
## 配置tomcat插件



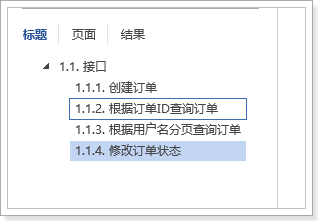
## 配置nginx



## 配置hosts



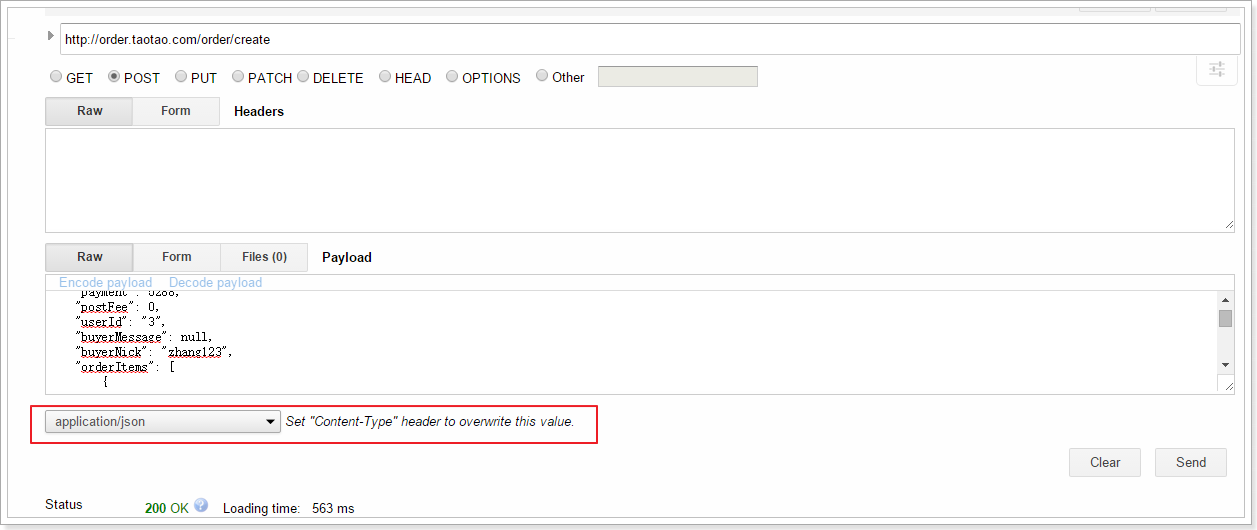
## 接口文档



## 接口

### 创建订单接口

使用：



### 如何生成订单号

订单号需求是什么？

1. 唯一
2. 可读性高
   1. 纯数字
3. 长度
   1. 不长于20位

如何生成：

1. 时间戳
   1. 可能重复
2. 手机号
3. 时间戳 + 随机数
   1. 可能重复
4. 时间戳 + 自增id
   1. 可行 （使用Redis的INCR命令完成）
5. 用户id + 时间戳
6. 用户id + 店铺id + 时间戳
   1. 可行，适用于C2C、B2C平台

### 具体实现

Controller： 接收提交的json数据

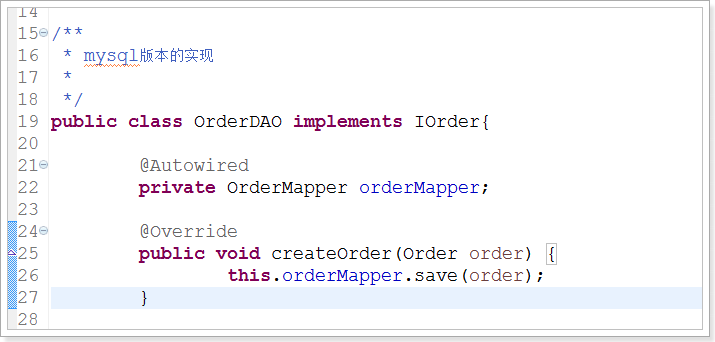


Service：将json反序列化为Order对象，对Order对象做校验，生成订单ID：

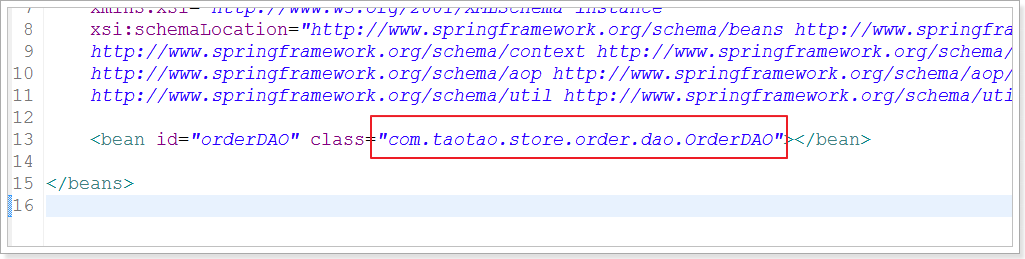




OrderDAO是一个接口的实现类，后期如果对订单数据的存储做改造时，保存订单就会有多套实现：

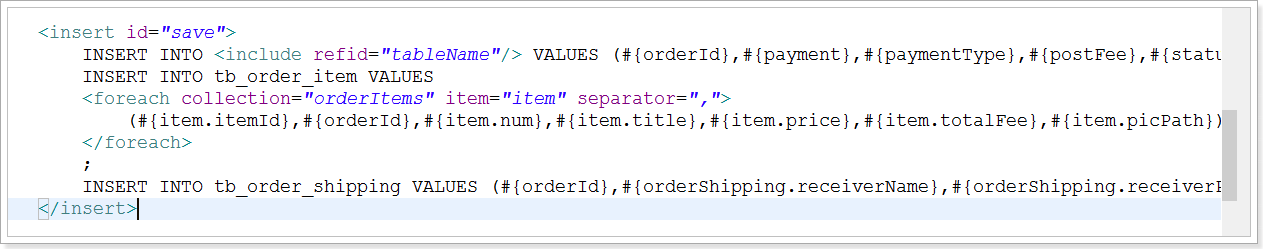


通过配置将OrderDAO加入到Spring容器：



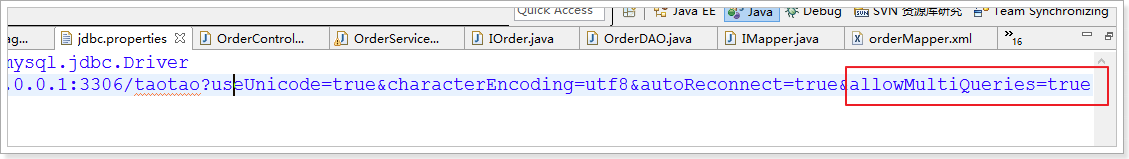
使用配置文件配置bean，后期的修改更加灵活。

Mapper.xml:



一个statement执行多条SQL语句，默认情况下支持吗？ -- 不支持的。

在连接字符串中设置：



是在同一个事务中吗？ -- 是的。

但是，现在的代码中存在一个bug，事务不会回滚，原因：Service中的异常被捕获：



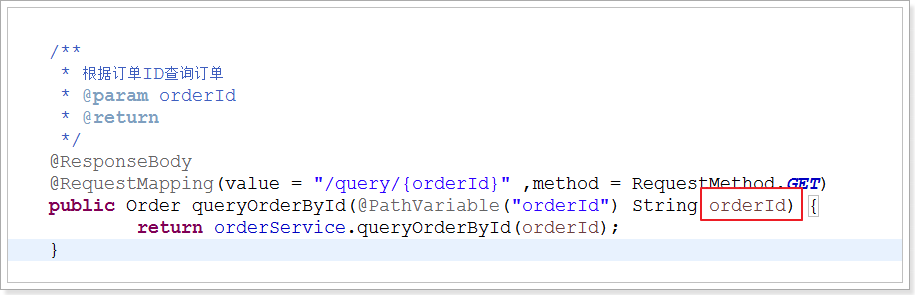
## 根据订单ID查询订单

### 测试

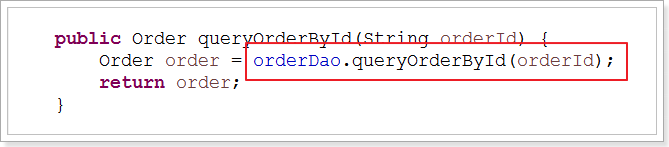




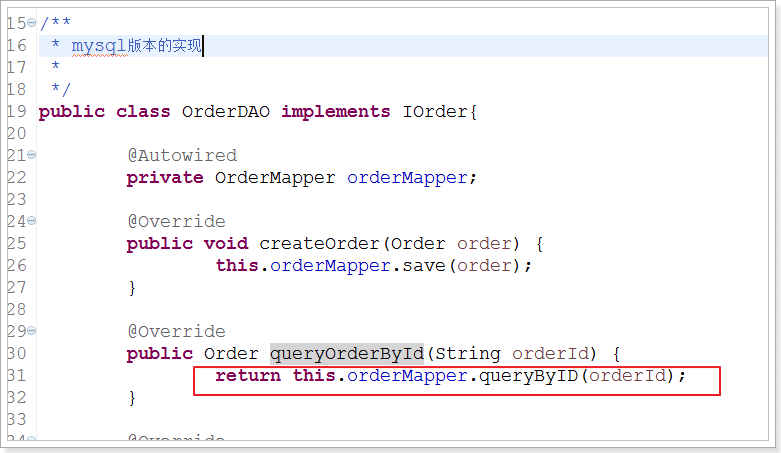
### Controller



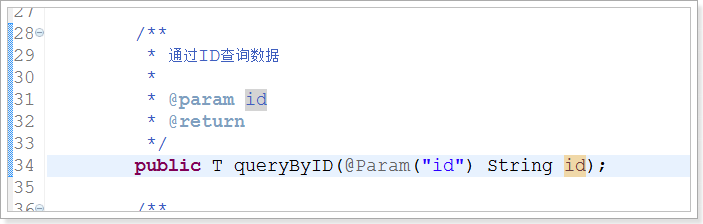
### Service



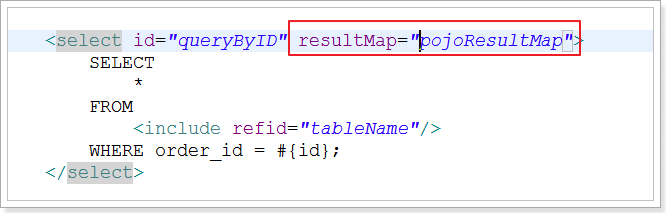
### OrderDAO



### OrderMapper

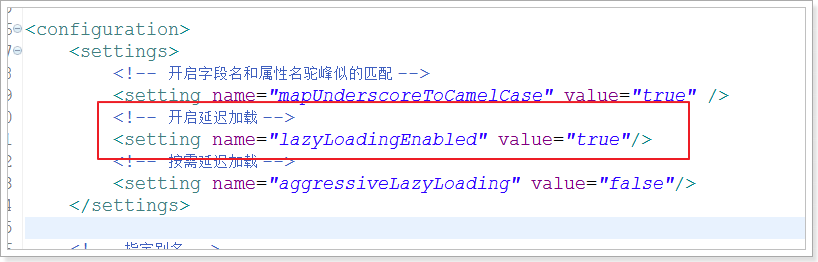


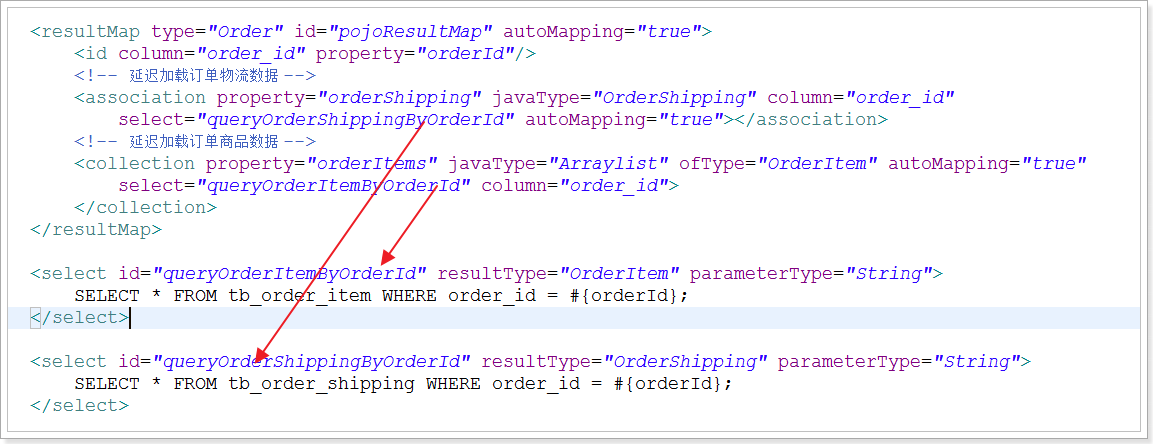
### OrderMapper.xml



通过**延迟加载**实现订单商品和订单物流数据的加载：

开启延迟加载：





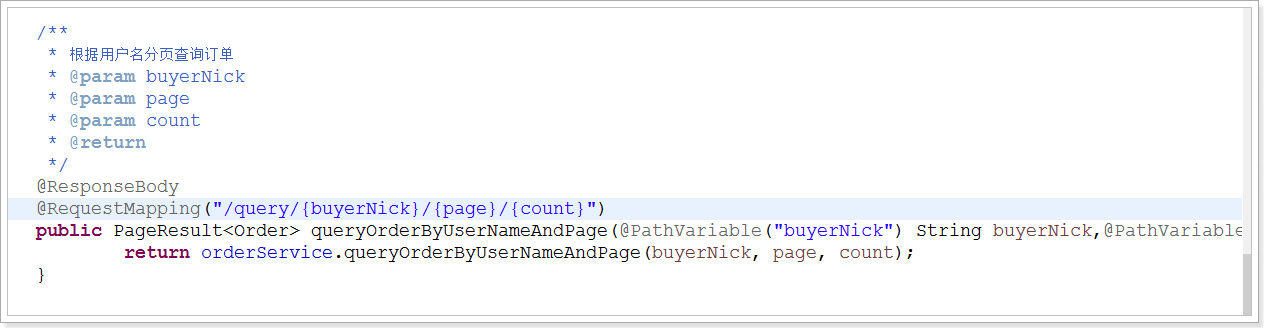
## 根据用户名分页查询订单

### 测试

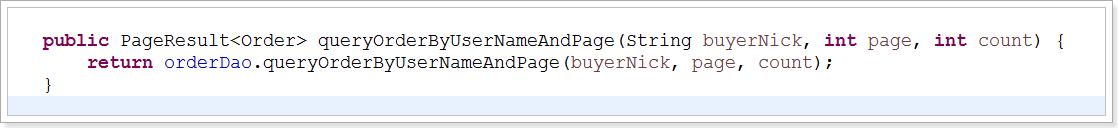




### Controller



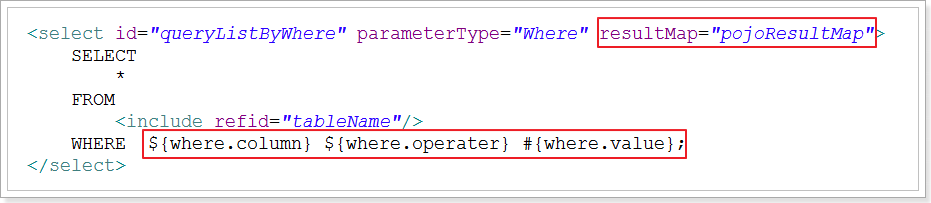
### Service



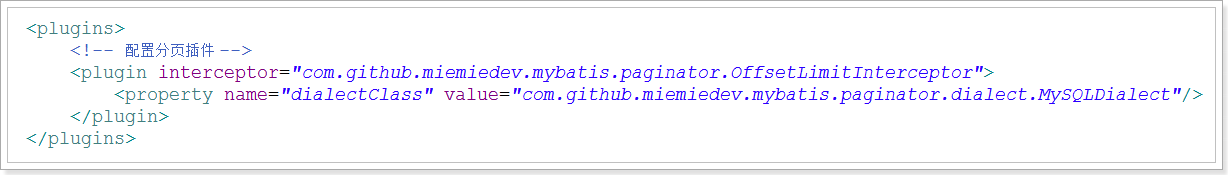
### OrderDAO



Mapp.xml：（使用延迟加载）



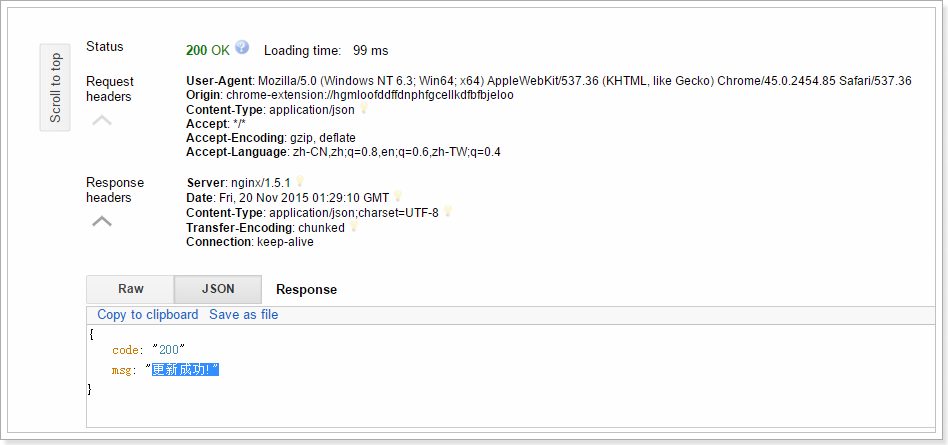
分页插件的配置（不要求掌握，了解即可）：



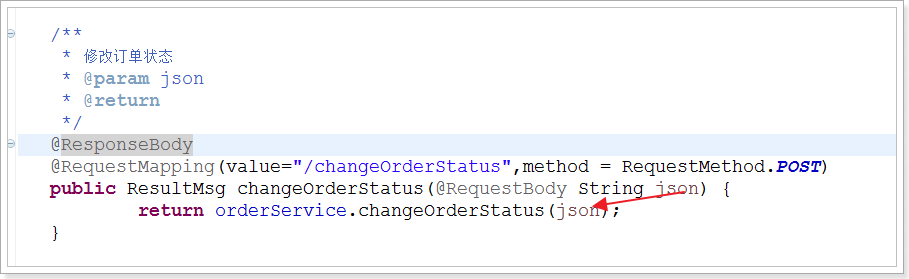
## 修改订单状态

### 测试



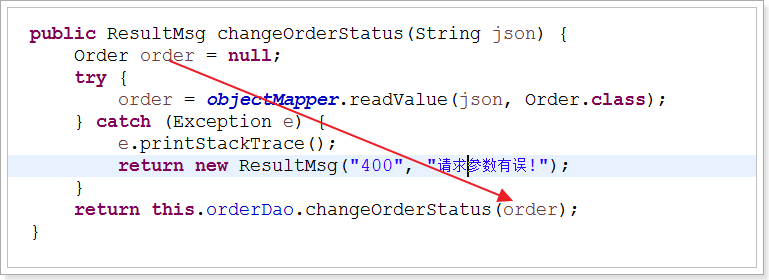


### Controller



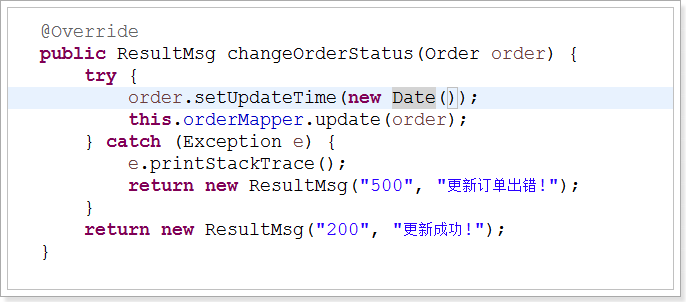
### Service

将json反序列化成Order对象：



### OrderDAO

设置更新时间：



### OrderMapper.xml

这是一个通用的更新实现：

<update id=*"update"*>

UPDATE <include refid=*"tableName"*/>

<set>

<if test=*"payment !=null and payment != ''"*>

payment = #{payment},

</if>

<if test=*"postFee !=null and postFee != ''"*>

post\_fee = #{postFee},

</if>

<if test=*"status !=null and status != ''"*>

status = #{status},

</if>

<if test=*"updateTime !=null and updateTime != ''"*>

update\_time = #{updateTime},

</if>

<if test=*"paymentTime !=null and paymentTime != ''"*>

payment\_time = #{paymentTime},

</if>

<if test=*"consignTime !=null and consignTime != ''"*>

consign\_time = #{consignTime},

</if>

<if test=*"endTime !=null and endTime != ''"*>

end\_time = #{endTime},

</if>

<if test=*"closeTime !=null and closeTime != ''"*>

close\_time = #{closeTime},

</if>

<if test=*"shippingName !=null and shippingName != ''"*>

shipping\_name = #{shippingName},

</if>

<if test=*"shippingCode !=null and shippingCode != ''"*>

shipping\_code = #{shippingCode},

</if>

<if test=*"buyerMessage !=null and buyerMessage != ''"*>

buyer\_message = #{buyerMessage},

</if>

<if test=*"buyerRate !=null and buyerRate != ''"*>

buyer\_rate = #{buyerRate},

</if>

</set>

WHERE order\_id = #{orderId};

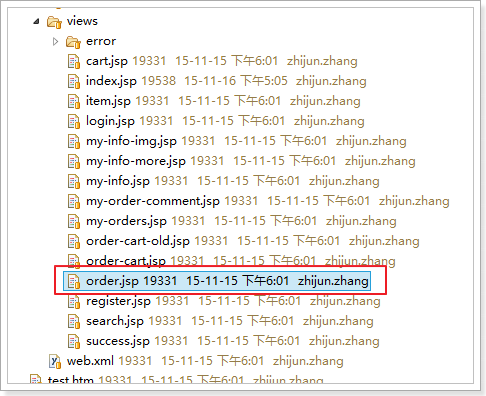
</update>

# 基于订单系统接口完成下单功能

## 下单按钮功能



## 订单确认页



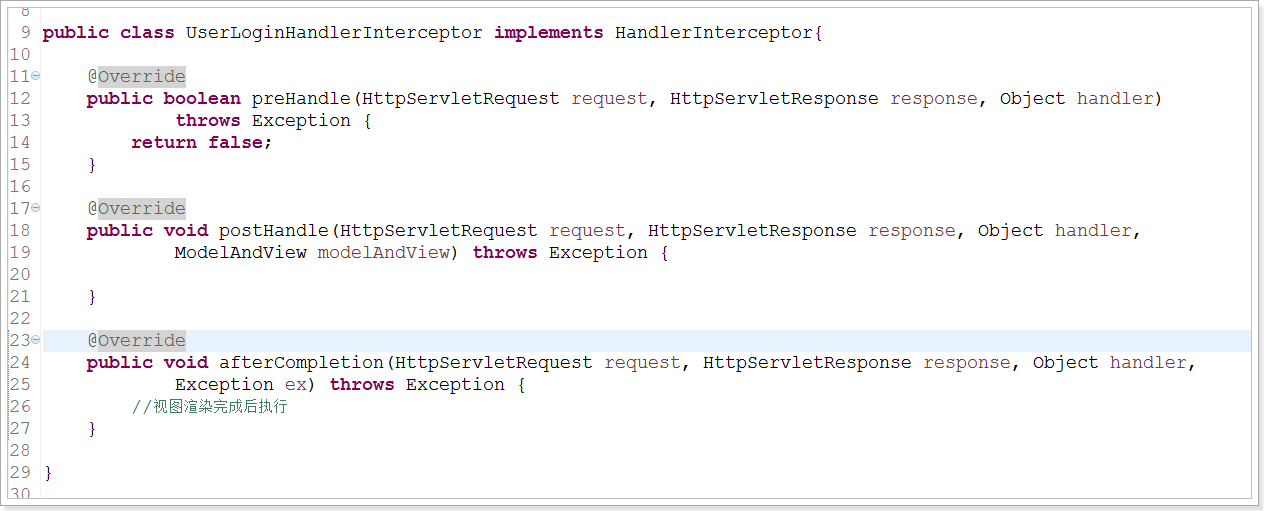
## 去订单确认页



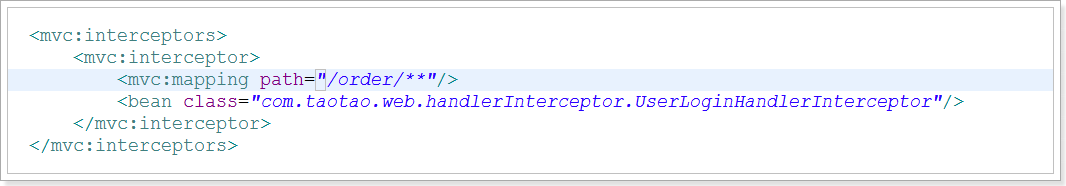


## 使用拦截器实现用户是否登录的校验

### 拦截器



### 配置拦截器



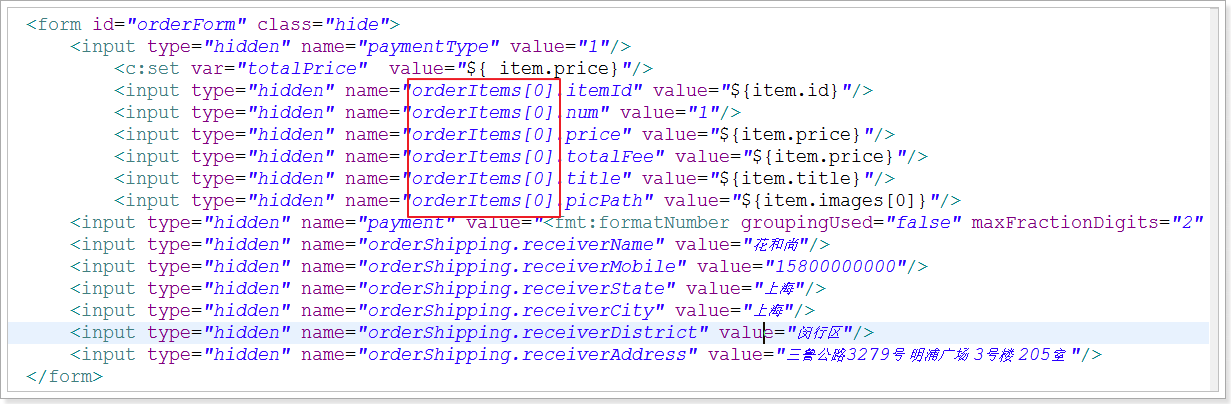
### 编写具体的业务逻辑



## 点击提交订单事件



表单数据：

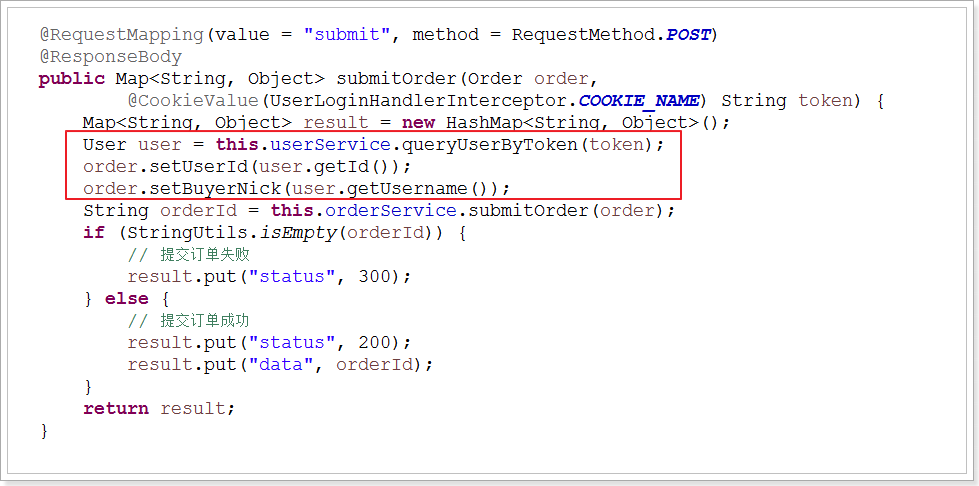


## 接收提交订单请求

### 在web.xml中添加service/\*请求进入SpringMVC



### Controller



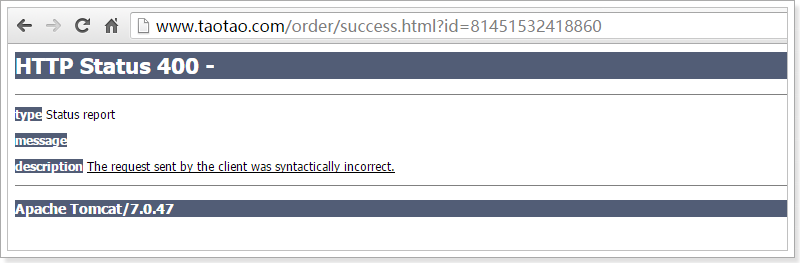
### Service



### 在ApiService中扩展doPostJson方法



### 测试



## 通过token进行了2次查询

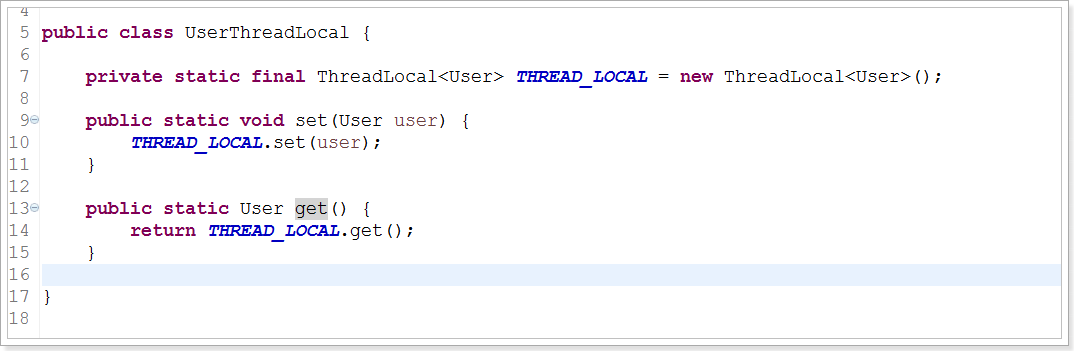
一次是在拦截器中查询，一次是在Controller查询，存在性能和资源浪费问题。

如何将拦截器中的数据传递到Controller？

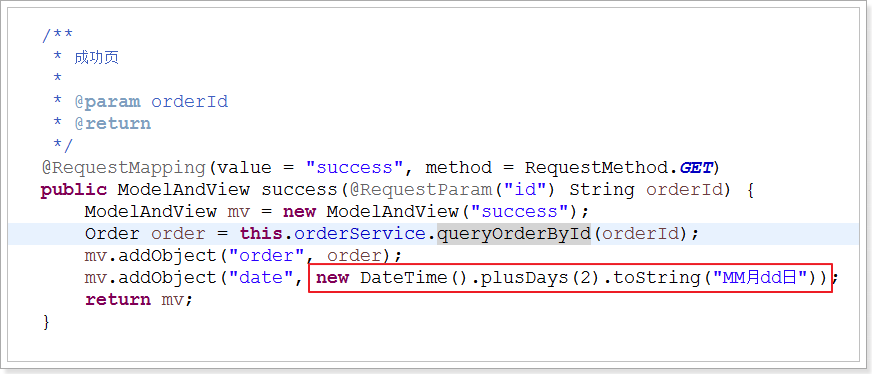
方案：

1. 将User对象放置到request对象中
2. 使用ThreadLocal实现
   1. 进入tomcat和产生响应前，都处于同一个线程中

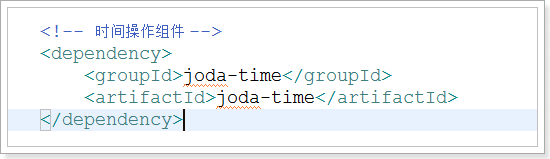
实现：

1. 定义ThreadLocal  
   
2. 在拦截器中将User对象放置到ThreadLocal中  
   
3. 在Controller或Service获取User对象  
   

## 成功页



使用joda-time组件：



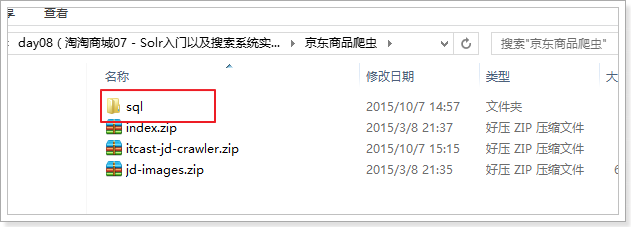
测试：

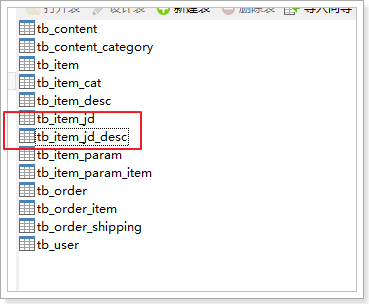


# 实现商品的搜索

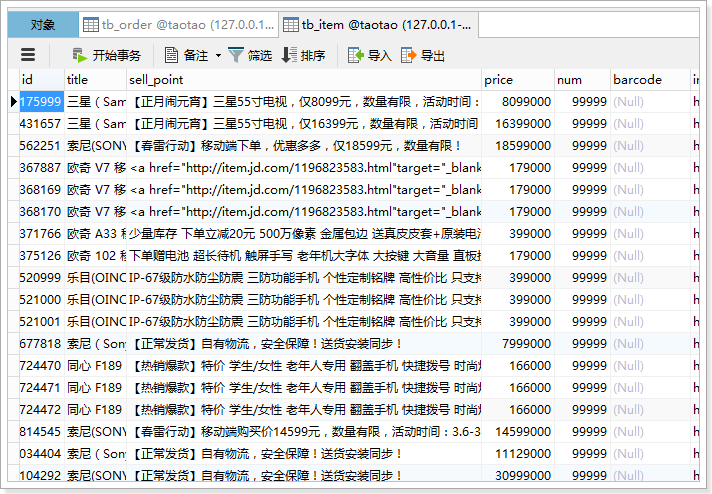
## 导入数据

数据来源，京东，在：

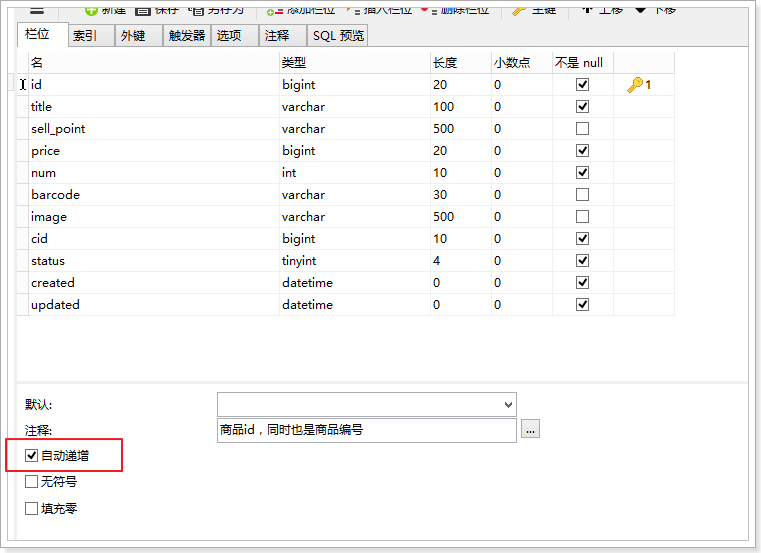




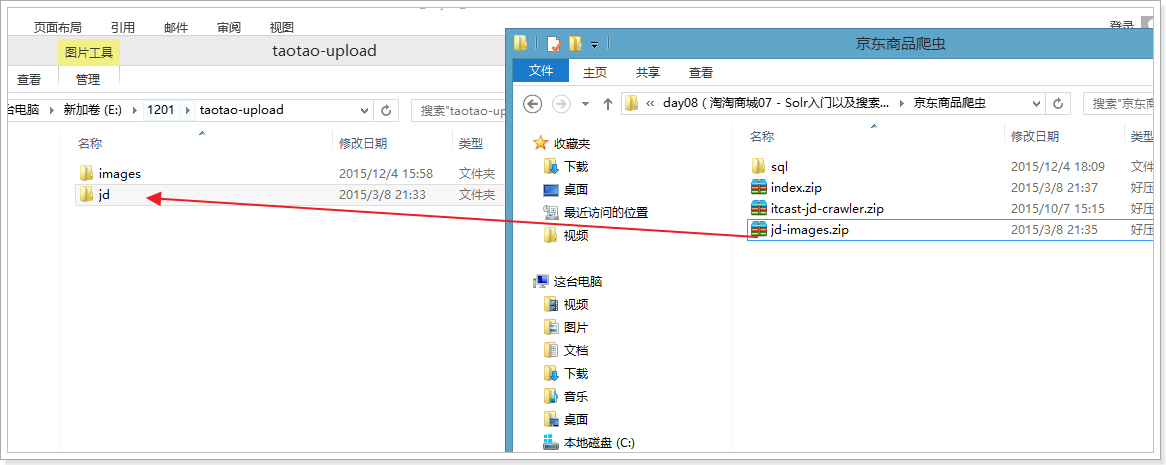
数据：



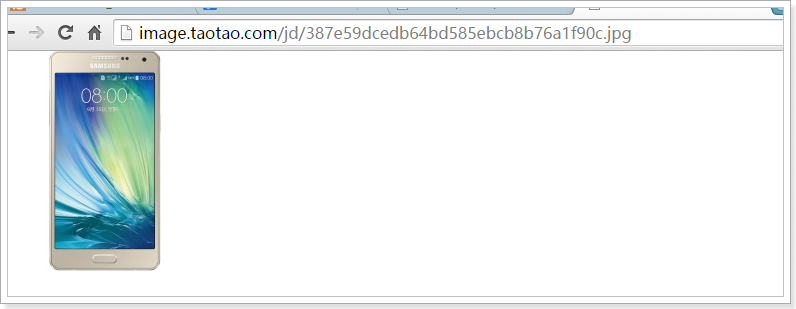
## 修改表结构



## 导入图片数据

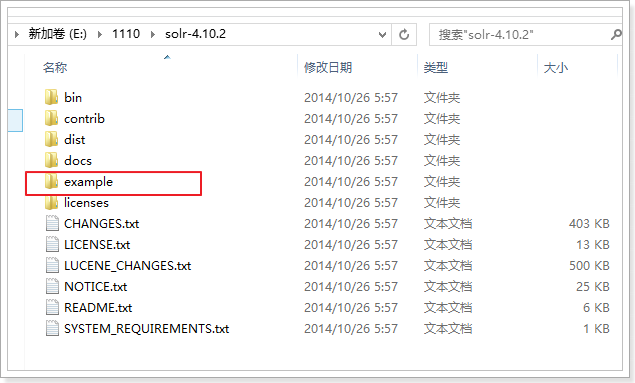


测试：



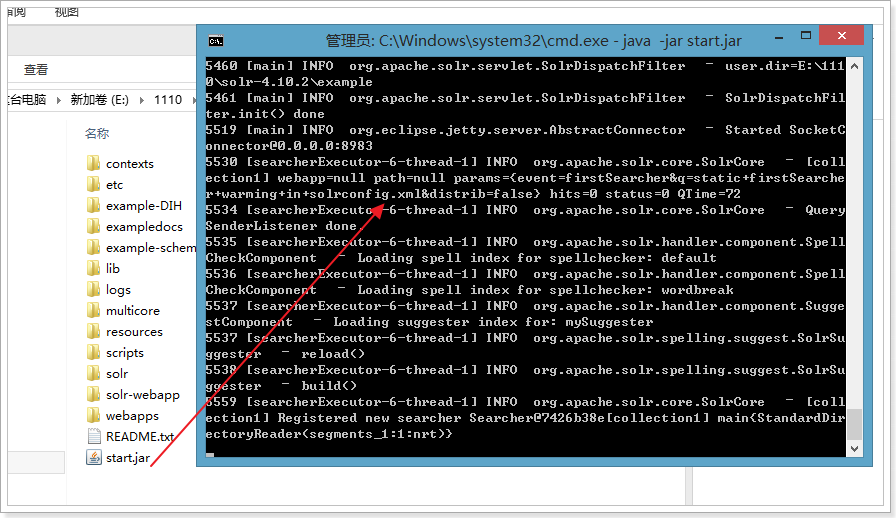
## Solr的安装

* 安装参考《Solr安装手册.docx》



启动solr：

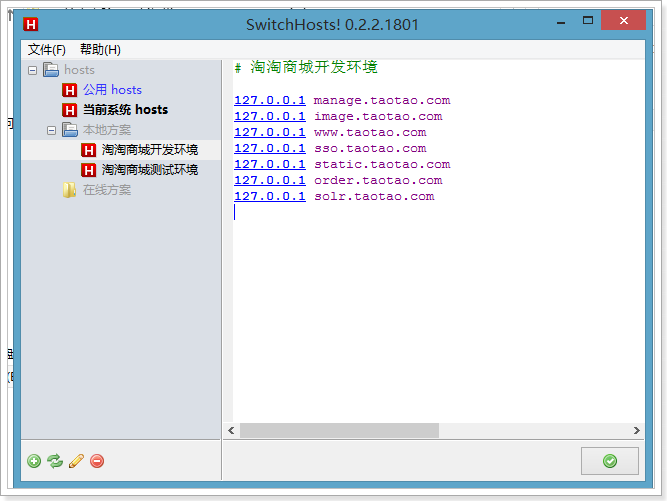
java -jar start.jar





## 通过域名访问solr服务

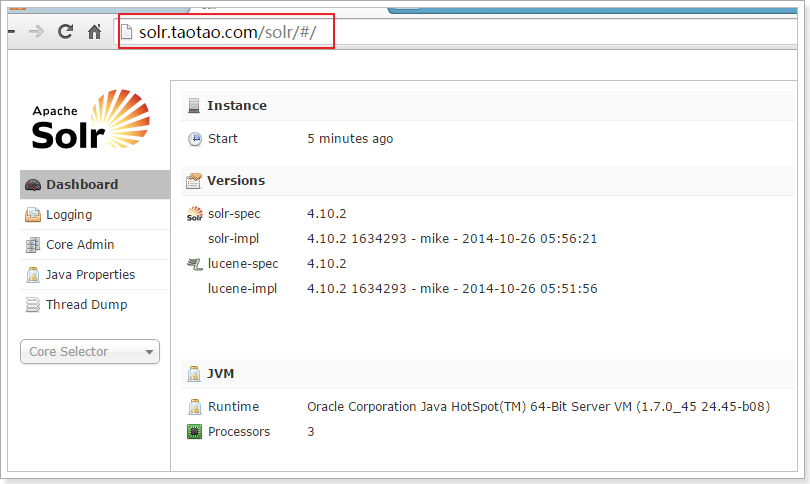
通过solr.taotao.com访问。



配置nginx：



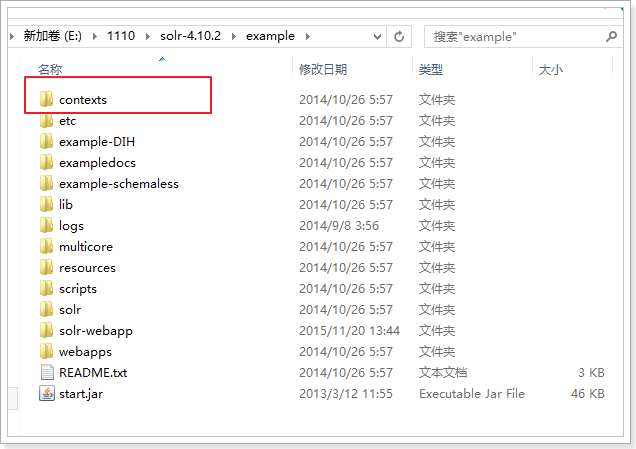
测试：

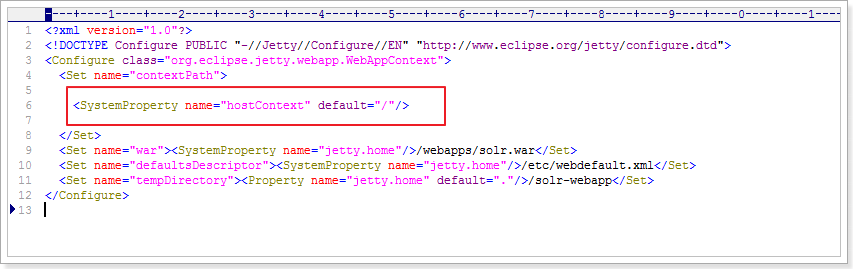


发现，请求域名后面依然得有/solr.

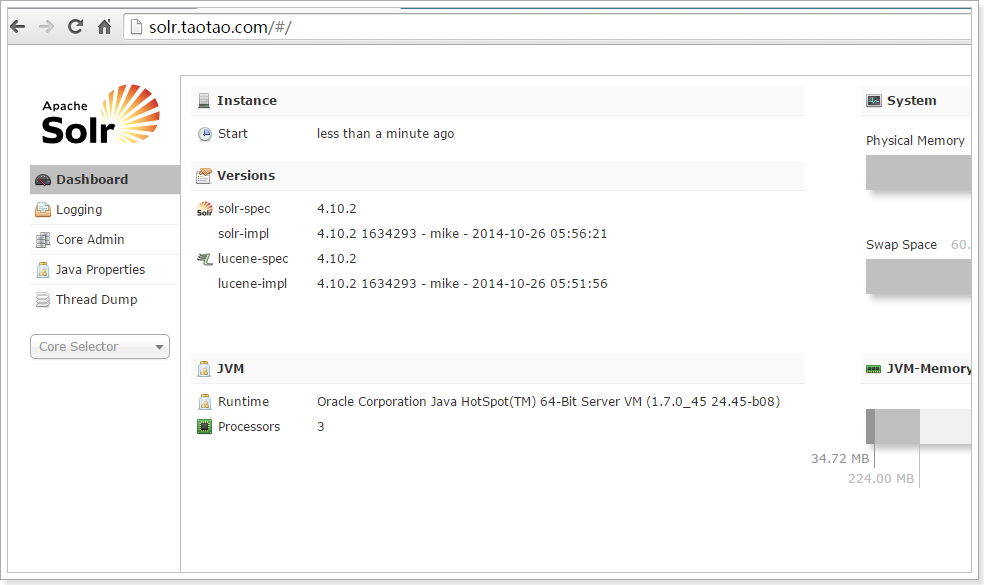
默认启动的web应用服务器是jetty。

修改配置文件：



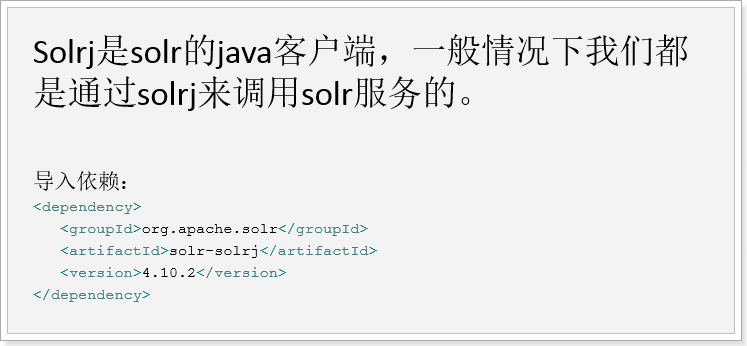


测试：



完美。

## Solrj



## 创建taotao core

严格按照该文档执行：具体参考《创建taotao core.docx 》

索引结构：



启动：

java -Dsolr.solr.home=taotao-solr -jar start.jar

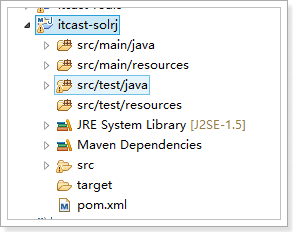
## 添加IK中文分词器的支持

参考该文档实现：《集成IKAnalyzer中文分词器.docx》

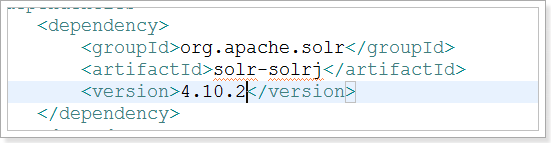


## 导入商品数据到solr中

### 导入itcast-solrj



### 导入依赖



### 使用Solrj完成索引数据的CRUD

**public** **class** ItemDataTest {

**private** HttpSolrServer httpSolrServer;

@Before

**public** **void** setUp() **throws** Exception {

// 在url中指定core名称：taotao

//http://solr.taotao.com/#/taotao -- 界面地址

String url = "http://solr.taotao.com/taotao"; //服务地址

HttpSolrServer httpSolrServer = **new** HttpSolrServer(url); //定义solr的server

httpSolrServer.setParser(**new** XMLResponseParser()); // 设置响应解析器

httpSolrServer.setMaxRetries(1); // 设置重试次数，推荐设置为1

httpSolrServer.setConnectionTimeout(500); // 建立连接的最长时间

**this**.httpSolrServer = httpSolrServer;

}

@Test

**public** **void** testInsert() **throws** Exception{

Item item = **new** Item();

item.setCid(1L);

item.setId(999L);

item.setImage("image");

item.setPrice(100L);

item.setSellPoint("很好啊，赶紧来买吧.");

item.setStatus(1);

item.setTitle("飞利浦 老人手机 (X2560) 深情蓝 移动联通2G手机 双卡双待");

**this**.httpSolrServer.addBean(item);

**this**.httpSolrServer.commit();

}

@Test

**public** **void** testUpdate() **throws** Exception{

Item item = **new** Item();

item.setCid(1L);

item.setId(999L);

item.setImage("image");

item.setPrice(100L);

item.setSellPoint("很好啊，赶紧来买吧. 豪啊");

item.setStatus(1);

item.setTitle("飞利浦 老人手机 (X2560) 深情蓝 移动联通2G手机 双卡双待");

**this**.httpSolrServer.addBean(item);

**this**.httpSolrServer.commit();

}

@Test

**public** **void** testDelete() **throws** Exception{

**this**.httpSolrServer.deleteById("999");

**this**.httpSolrServer.commit();

}

@Test

**public** **void** testQuery() **throws** Exception{

**int** page = 2;

**int** rows = 1;

String keywords = "手机";

SolrQuery solrQuery = **new** SolrQuery(); //构造搜索条件

solrQuery.setQuery("title:" + keywords); //搜索关键词

// 设置分页 start=0就是从0开始，，rows=5当前返回5条记录，第二页就是变化start这个值为5就可以了。

solrQuery.setStart((Math.*max*(page, 1) - 1) \* rows);

solrQuery.setRows(rows);

//是否需要高亮

**boolean** isHighlighting = !StringUtils.*equals*("\*", keywords) && StringUtils.*isNotEmpty*(keywords);

**if** (isHighlighting) {

// 设置高亮

solrQuery.setHighlight(**true**); // 开启高亮组件

solrQuery.addHighlightField("title");// 高亮字段

solrQuery.setHighlightSimplePre("<em>");// 标记，高亮关键字前缀

solrQuery.setHighlightSimplePost("</em>");// 后缀

}

// 执行查询

QueryResponse queryResponse = **this**.httpSolrServer.query(solrQuery);

List<Item> items = queryResponse.getBeans(Item.**class**);

**if** (isHighlighting) {

// 将高亮的标题数据写回到数据对象中

Map<String, Map<String, List<String>>> map = queryResponse.getHighlighting();

**for** (Map.Entry<String, Map<String, List<String>>> highlighting : map.entrySet()) {

**for** (Item item : items) {

**if** (!highlighting.getKey().equals(item.getId().toString())) {

**continue**;

}

item.setTitle(StringUtils.*join*(highlighting.getValue().get("title"), ""));

**break**;

}

}

}

**for** (Item item : items) {

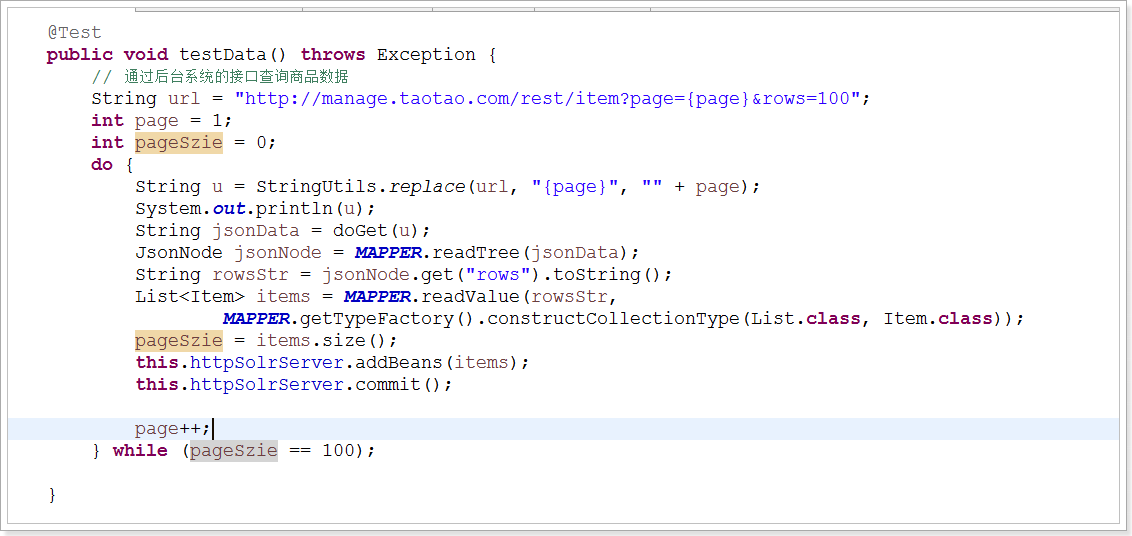
System.***out***.println(item);

}

}

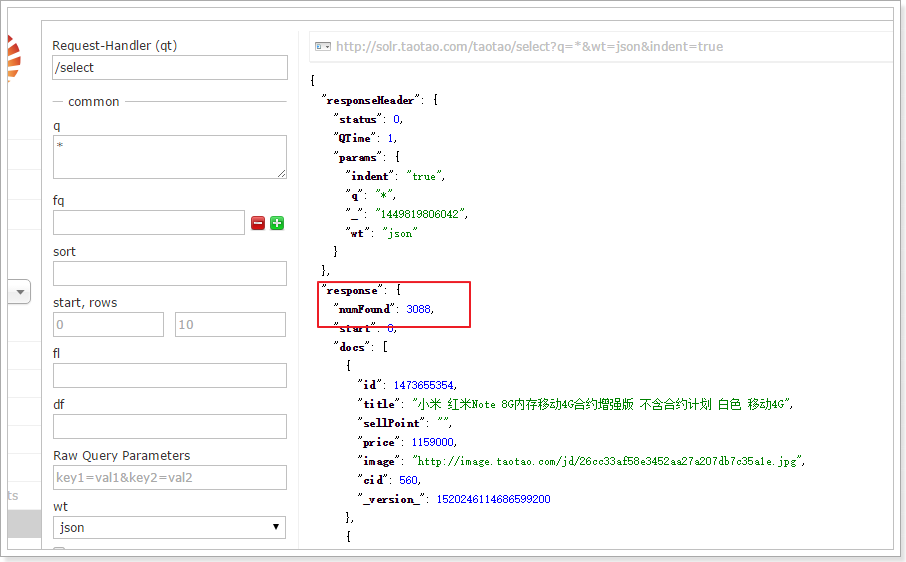
}

### 将商品数据到入到solr





测试：



OK。

# 创建搜索系统taotao-search

## 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.search</groupId>

<artifactId>taotao-search</artifactId>

<version>1.0.0-SNAPSHOT</version>

<packaging>war</packaging>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</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>

<!-- 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>

<dependency>

<groupId>org.apache.solr</groupId>

<artifactId>solr-solrj</artifactId>

<version>4.10.2</version>

</dependency>

</dependencies>

<build>

<plugins>

<!-- 配置Tomcat插件 -->

<plugin>

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

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

<configuration>

<port>8085</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-search</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-search</servlet-name>

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

<init-param>

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

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

</init-param>

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

</servlet>

<servlet-mapping>

<servlet-name>taotao-search</servlet-name>

<!--

伪静态

伪静态有利于SEO（搜索引擎优化）

-->

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

</servlet-mapping>

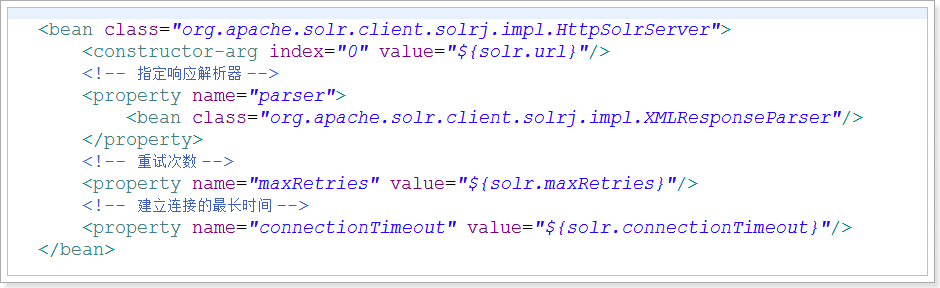
<welcome-file-list>

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

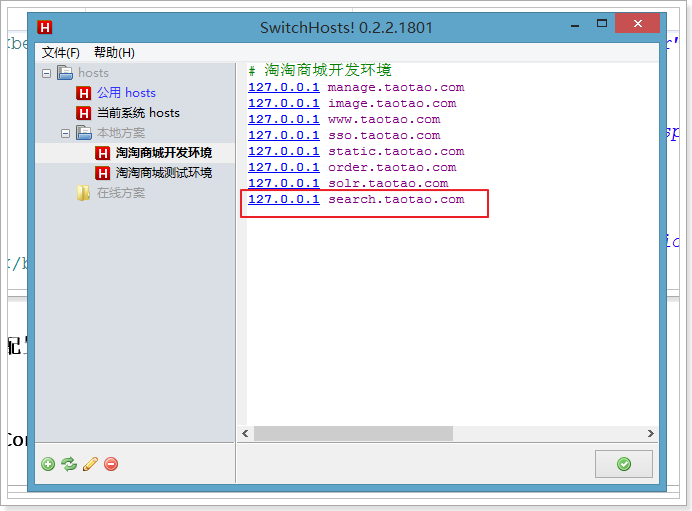
</welcome-file-list>

</web-app>

## Solr和Spring的整合



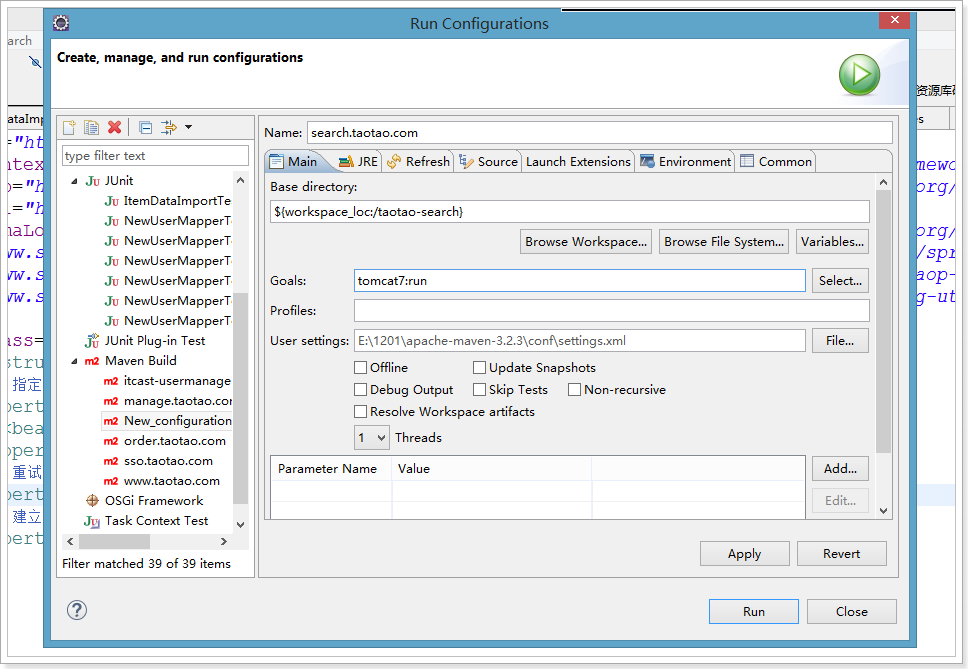
## 配置nginx和hosts



配置nginx：



配置启动tomcat：



## Controller

@RequestMapping(value = "search", method = RequestMethod.***GET***)

**public** ModelAndView search(@RequestParam("q") String keyWords,

@RequestParam(value = "page", defaultValue = "1") Integer page) {

ModelAndView mv = **new** ModelAndView("search");

SearchResult searchResult = **null**;

**try** {

searchResult = **this**.searchService.search(keyWords, page, ***ROWS***);

} **catch** (SolrServerException e) {

e.printStackTrace();

searchResult = **new** SearchResult(**new** ArrayList<Item>(0), 0L);

}

// 搜索关键字

mv.addObject("query", keyWords);

// 搜索结果集

mv.addObject("itemList", searchResult.getList());

// 当前页数

mv.addObject("page", page);

// 总页数

**int** total = searchResult.getTotal().intValue();

**int** pages = total % ***ROWS*** == 0 ? total / ***ROWS*** : total / ***ROWS*** + 1;

mv.addObject("pages", pages);

**return** mv;

}

## Service

**public** SearchResult searchItem(String keyWords, **int** page, **int** rows) {

SolrQuery solrQuery = **new** SolrQuery(); // 构造搜索条件

solrQuery.setQuery("title:" + keyWords + " AND status:1"); // 搜索关键词

// 设置分页 start=0就是从0开始，，rows=5当前返回5条记录，第二页就是变化start这个值为5就可以了。

solrQuery.setStart((Math.*max*(page, 1) - 1) \* rows);

solrQuery.setRows(rows);

// 是否需要高亮

**boolean** isHighlighting = !StringUtils.*equals*("\*", keyWords) && StringUtils.*isNotEmpty*(keyWords);

**if** (isHighlighting) {

// 设置高亮

solrQuery.setHighlight(**true**); // 开启高亮组件

solrQuery.addHighlightField("title");// 高亮字段

solrQuery.setHighlightSimplePre("<em>");// 标记，高亮关键字前缀

solrQuery.setHighlightSimplePost("</em>");// 后缀

}

**try** {

// 执行查询

QueryResponse queryResponse = **this**.httpSolrServer.query(solrQuery);

List<Item> items = queryResponse.getBeans(Item.**class**);

**if** (isHighlighting) {

// 将高亮的标题数据写回到数据对象中

Map<String, Map<String, List<String>>> map = queryResponse.getHighlighting();

**for** (Map.Entry<String, Map<String, List<String>>> highlighting : map.entrySet()) {

**for** (Item item : items) {

**if** (!highlighting.getKey().equals(item.getId().toString())) {

**continue**;

}

item.setTitle(StringUtils.*join*(highlighting.getValue().get("title"), ""));

**break**;

}

}

}

**return** **new** SearchResult(queryResponse.getResults().getNumFound(), items);

} **catch** (SolrServerException e) {

e.printStackTrace();

}

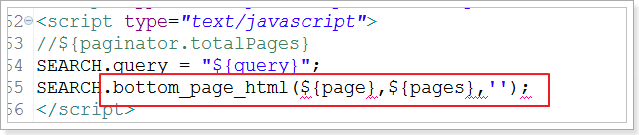
**return** **null**;

}

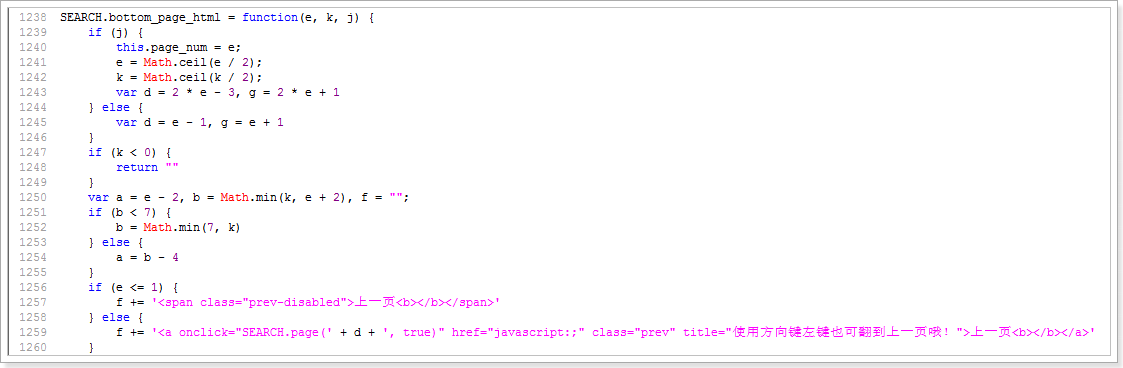
## 测试

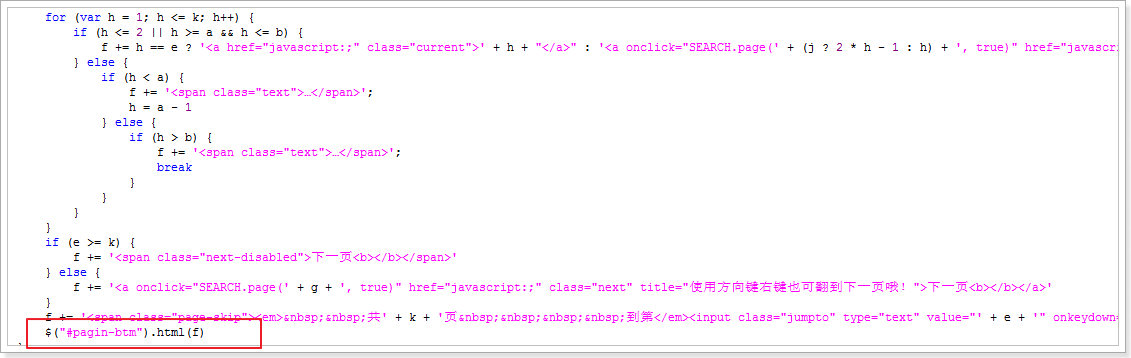


## 分页JS逻辑



具体方法：

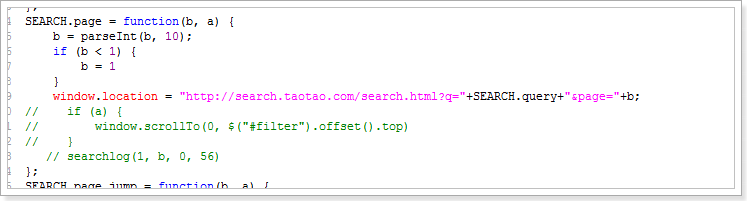




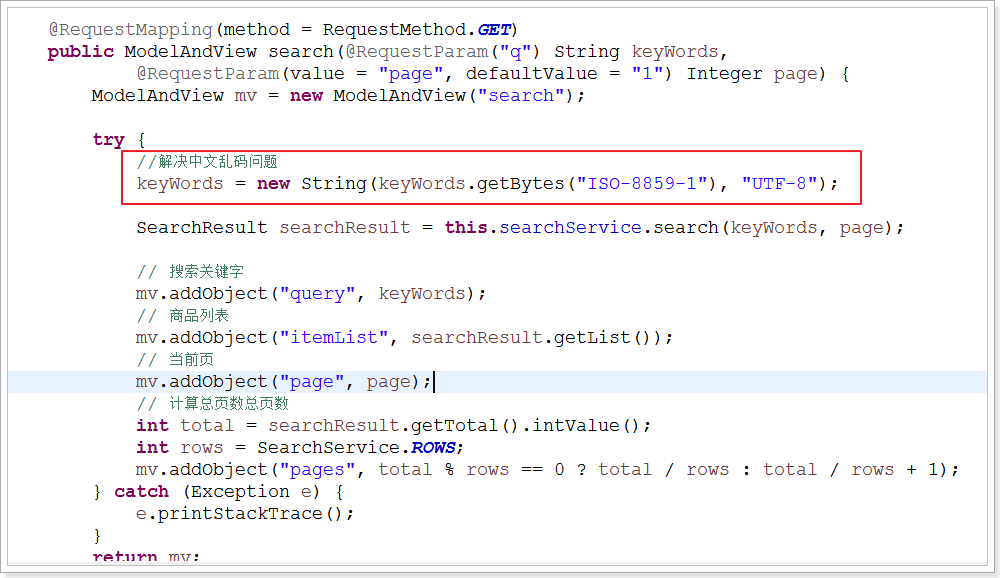
效果：



下一页的逻辑：



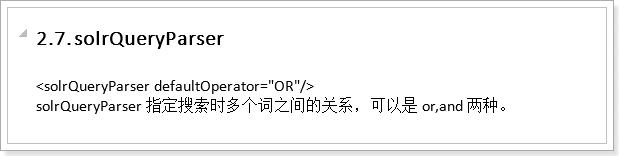
## 解决中文乱码问题

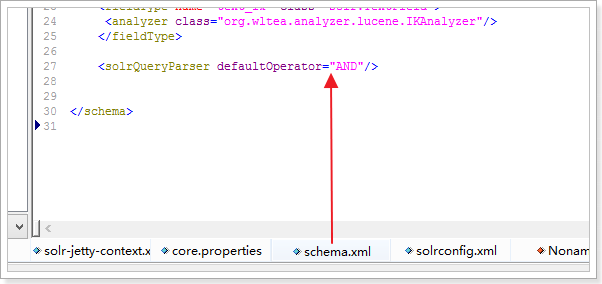


测试：



## 多词搜索的逻辑关系





测试：

