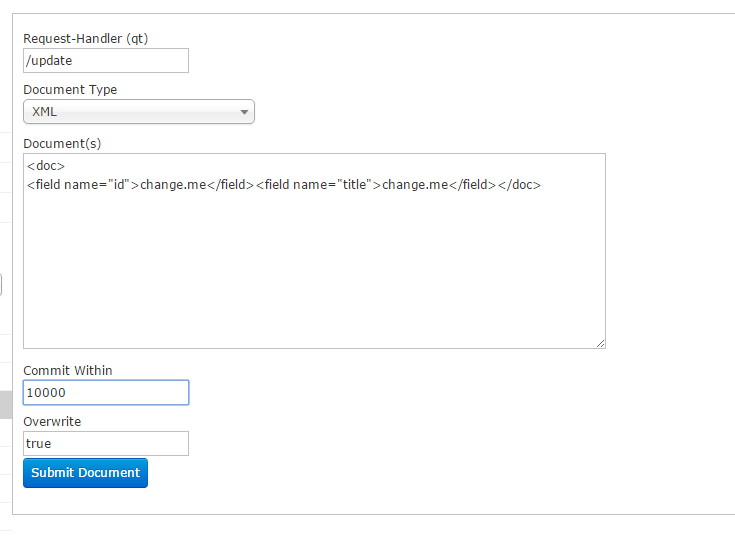
## 首先创建一个 solr 项目

<https://github.com/Ja0ck5/solr-practice/blob/master/doc/create-solr-core.docx>

## 删除数据

<delete><query>\*:\*</query></delete>

<commit/>



## Jeecg 排除冲突 jar 包

打开项目 pom.xml

形如一下排除冲突 jar 包，因为 solrj

<dependency>

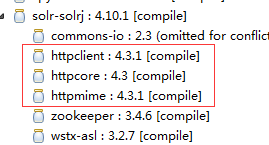
<groupId>org.apache.solr</groupId>

<artifactId>solr-solrj</artifactId>

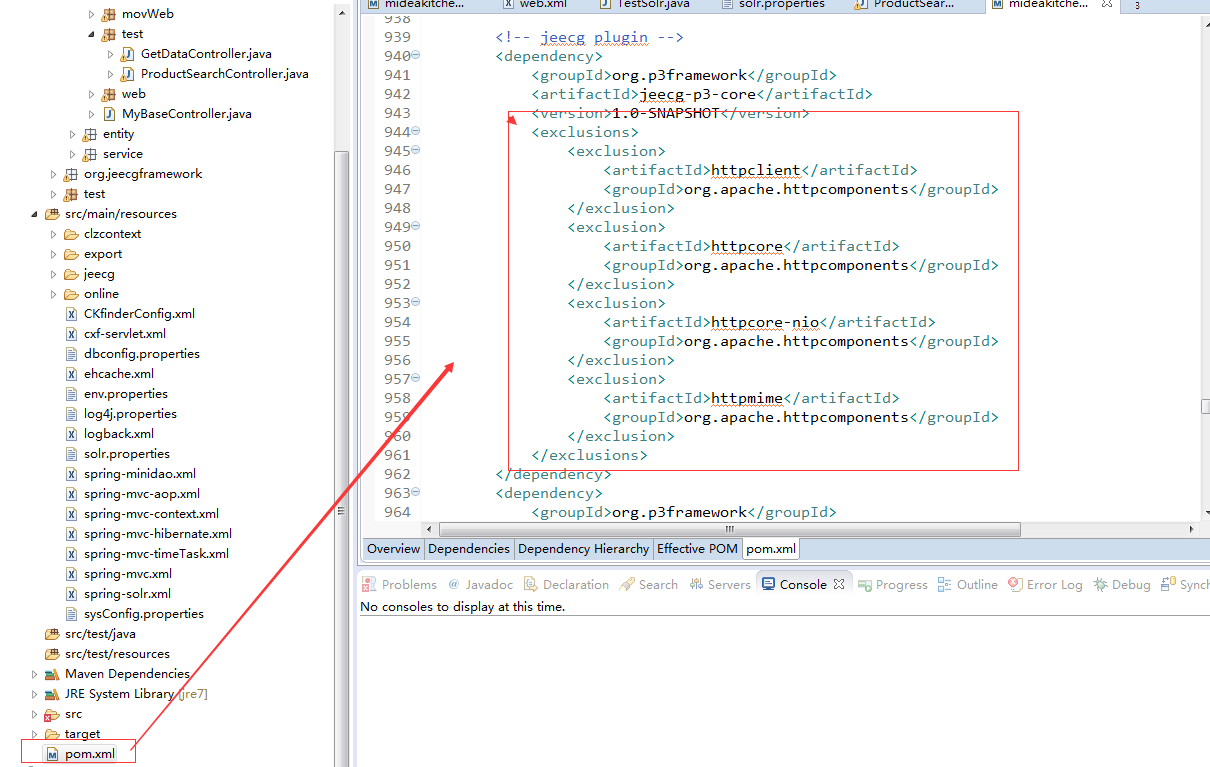
<version>4.10.1</version>

</dependency>

本身就有将 httpclient 依赖进来。但是jeecg 本身也有，不过版本太低，需要排除掉，使用 solrj 本身依赖的版本



形如一下排除，但是数量较多怎么办？

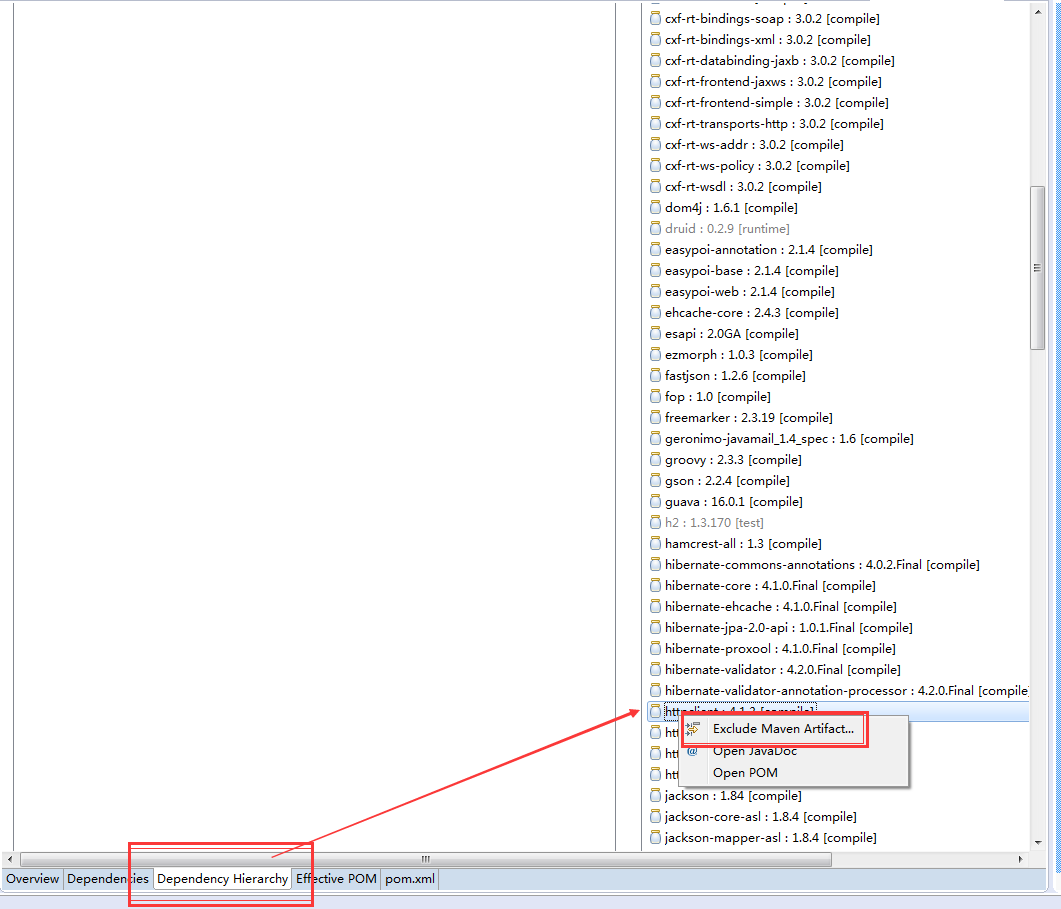


参考如下：

找到依赖树，找到版本 4.1.3 的 httpclient/httpcore/httpmine/httpcore-nio

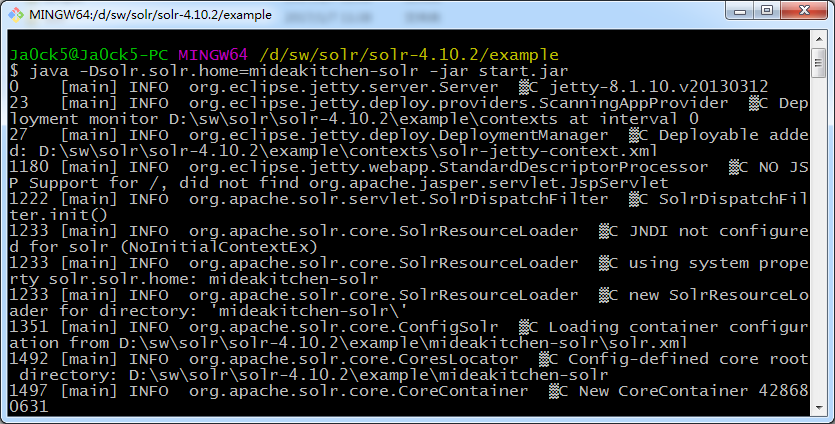
然后右键 Exclude Maven Artifact

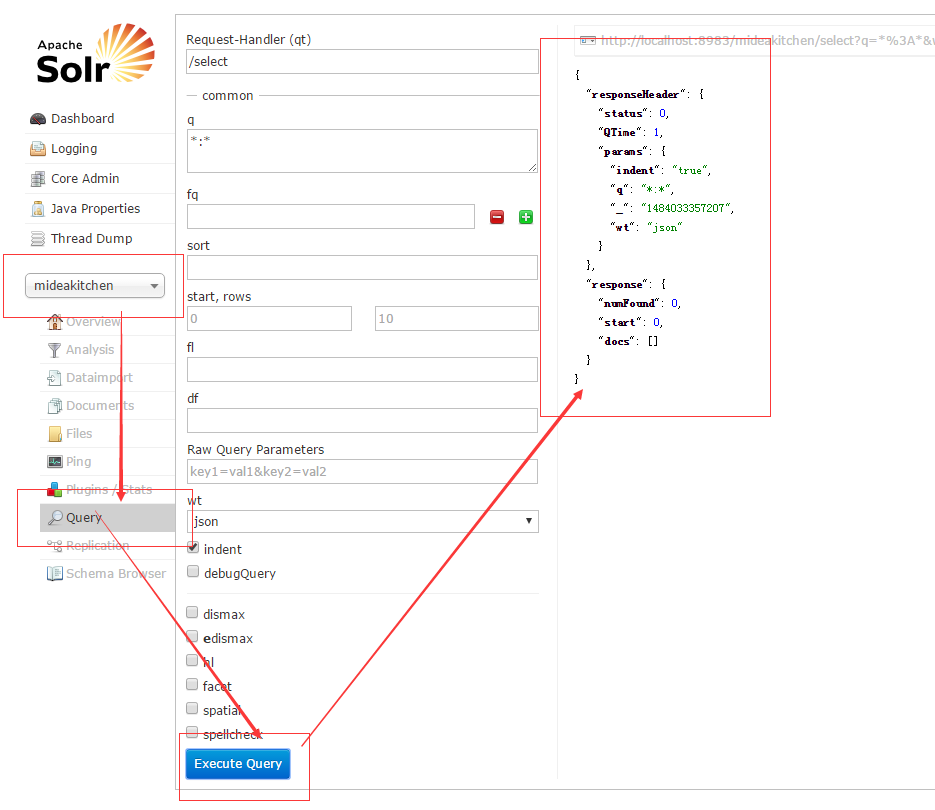
就可以排除掉不需要的版本，并且使用 solorj 自带的。



## 启动项目

$ java -Dsolr.solr.home=mideakitchen-solr -jar start.jar





## 开启后，导入数据

**public** Integer testSolr() **throws** Exception {

//http://solr.mallmall.com/#/mallmall 后台管理网页地址

String solrUrl = "http://localhost:8983/mideakitchen";//solr 服务地址

HttpSolrServer httpSolrServer = **new** HttpSolrServer(solrUrl); //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

//分页获取数据

Integer page = 1;

Integer pageSize = 100;

Integer dataSize = 0;

//将商品数据写入到solr

**do** {

List<SearchResult> searchResults = **new** ArrayList<>();

List<Map<String, Object>> searchResultDatas = **this**.productService.getSearchResult(page, pageSize);

dataSize = searchResultDatas.size();

**for** (Map<String, Object> map : searchResultDatas) {

SearchResult searchResult = **new** SearchResult();

searchResult.setProductId((String) map.get("productId"));

searchResult.setParentId((String) map.get("parentId"));

searchResult.setProductCategoryId((String) map.get("productCategoryId"));

searchResult.setProductType((Integer) map.get("productType"));

searchResult.setModel((String) map.get("model"));

searchResult.setDimensionalDrawing((String) map.get("dimensionalDrawing"));

searchResult.setStarLevel((Integer) map.get("starLevel"));

searchResult.setFeatureId((String) map.get("featureId"));

searchResult.setFeatures((String) map.get("features"));

searchResult.setMediaId((String) map.get("mediaId"));

searchResult.setMediaType((Integer) map.get("mediaType"));

searchResult.setOriginalUrl((String) map.get("originalUrl"));

searchResult.setThumbnailUrl((String) map.get("thumbnailUrl"));

searchResult.setUpdateDate((Date) map.get("updateDate"));

searchResults.add(searchResult);

}

httpSolrServer.addBeans(searchResults);

httpSolrServer.commit();

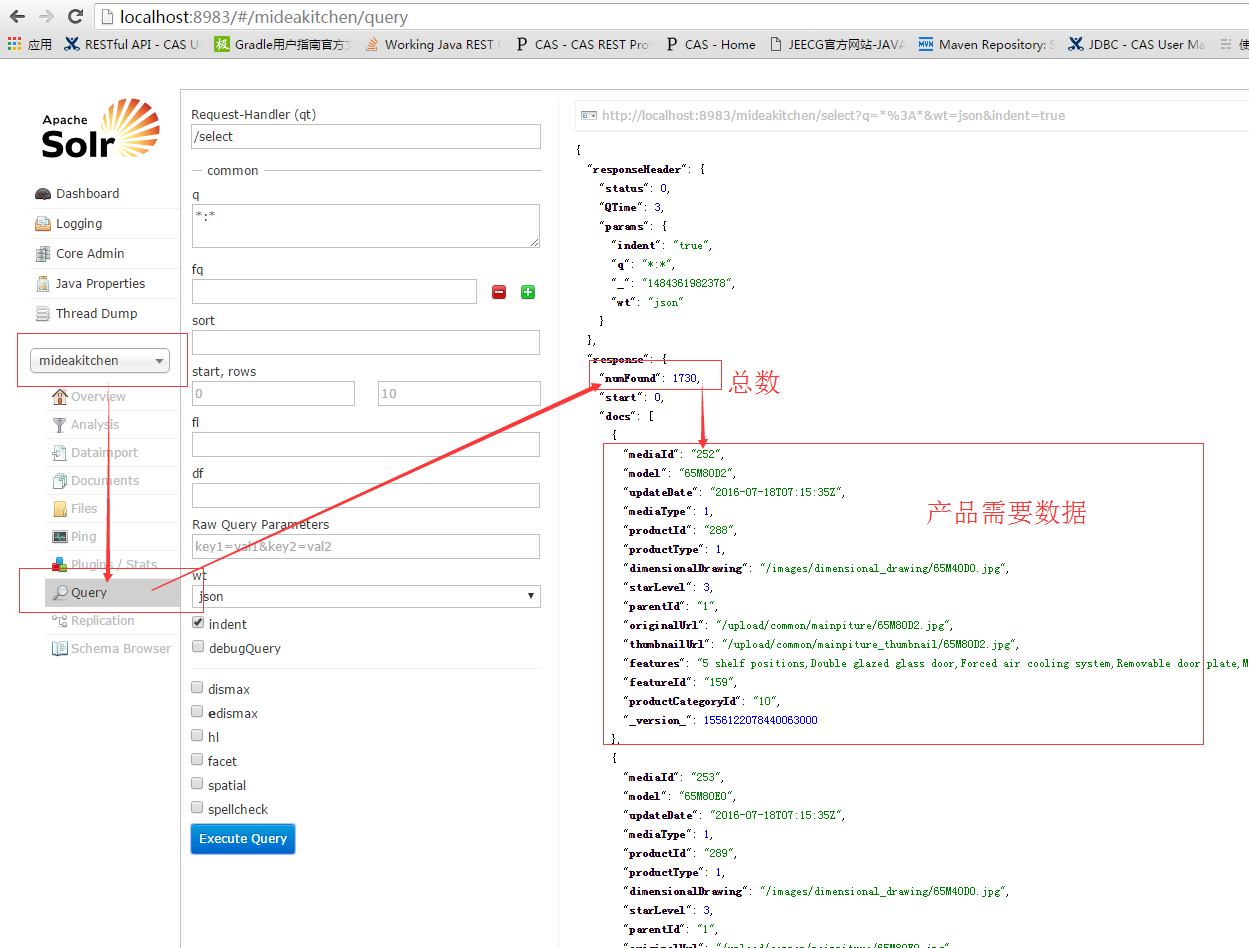
page++;

} **while** (100 == dataSize);

**return** page;

}

## 查看数据是否加入



## 查询数据并返回数据到前端

/\*\*

\*

\* productSearchController.do?search

\* **@param** req

\* **@param** rep

\* **@param** keyword

\* **@param** page

\* **@return**

\*/

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

@ResponseBody

**public** JsonResult search(HttpServletRequest req, HttpServletResponse rep, @RequestParam("keyword") String keyword,

@RequestParam("page") Integer page) {

JsonResult jr = **new** JsonResult();

**try** {

// **TODO** 分词问题

Map<String, Object> jsonMap = **new** HashMap<>();

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

solrQuery.setQuery("model:" + keyword); // 搜索关键词

// 设置分页 start=0就是从0开始，，rows=12当前返回12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*("\*", keyword) && StringUtils.*isNotEmpty*(keyword);

**if** (isHighlighting) {

// 设置高亮

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

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

solrQuery.setHighlightSimplePre("<em class=\"hight\_light\_model\">");// 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<SearchResult> searchResults = queryResponse.getBeans(SearchResult.**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** (SearchResult sr : searchResults) {

**if** (!highlighting.getKey().equals(sr.getProductId())) {

**continue**;

}

sr.setModel(StringUtils.*join*(highlighting.getValue().get("model"), ""));

**break**;

}

}

}

// 用户收藏夹(喜爱)

WUserEntity user = (WUserEntity) req.getSession().getAttribute("WUser");

**if** (user != **null**) {

List<UserFavoriteEntity> userFavoriteList = **this**.systemService.findByProperty(UserFavoriteEntity.**class**,"userId", user.getId());

jsonMap.put("userFavorites", userFavoriteList);

}

// 查询到的产品信息

jsonMap.put("searchResult", searchResults);

// 查询得到的总数量

jsonMap.put("total", queryResponse.getResults().size());

// 当前页

jsonMap.put("page", page);

jr.setResultCode(0);

jr.setResultMessage("OK");

jr.setResultObject(jsonMap);

} **catch** (Exception e) {

jr.setResultCode(2);

jr.setResultMessage(e.toString());

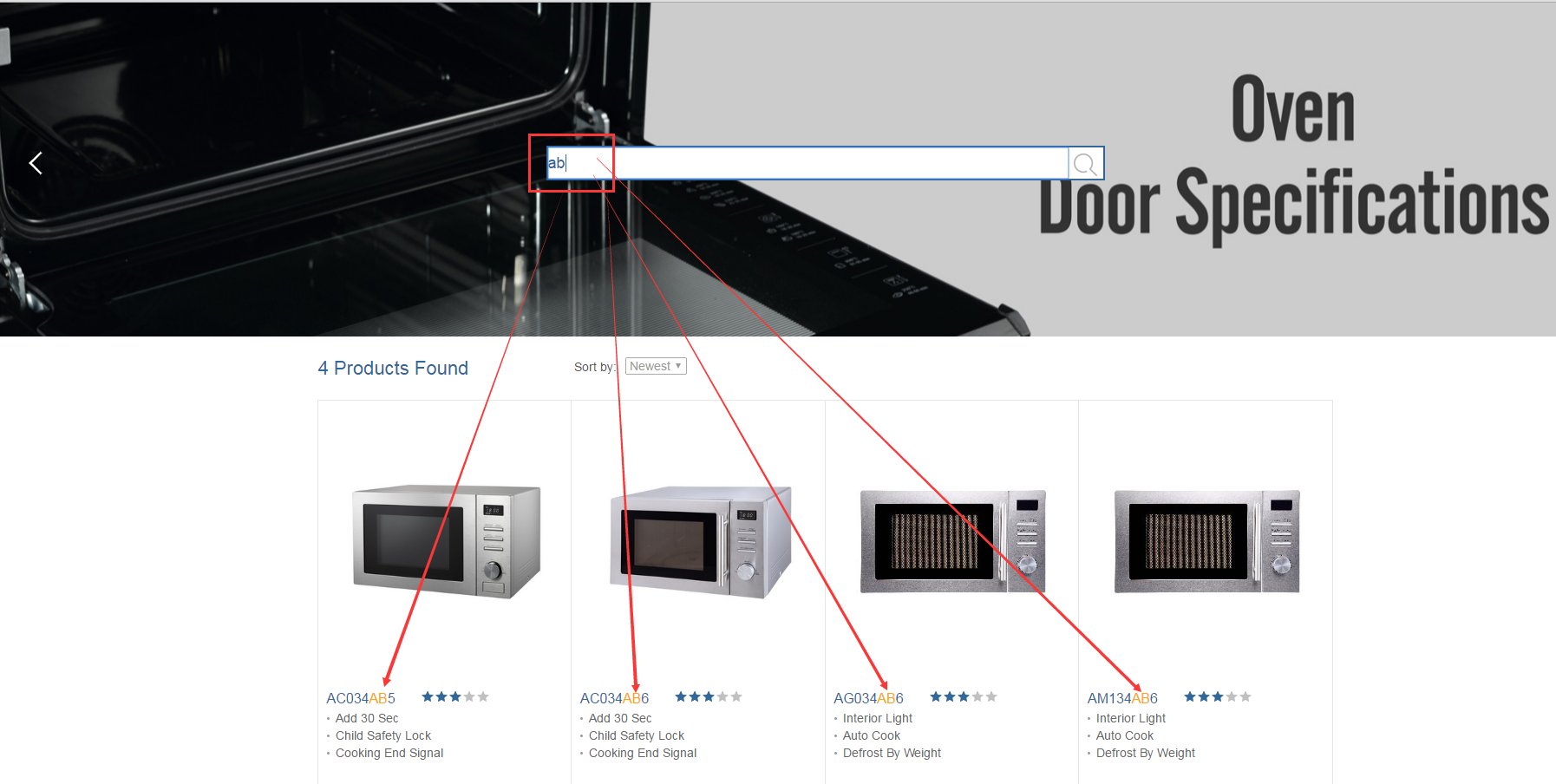
e.printStackTrace();

}

**return** jr;

}

前端查询数据：







## 配置文件：

solr.properties

solr.server.baseUrl=http://127.0.0.1:8983/mideakitchen

solr.maxRetries=1

solr.connectionTimeout=500

spring-solr.xml

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

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:mvc=*"http://www.springframework.org/schema/mvc"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns:p=*"http://www.springframework.org/schema/p"* xmlns:context=*"http://www.springframework.org/schema/context"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans-3.0.xsd*

*http://www.springframework.org/schema/context*

*http://www.springframework.org/schema/context/spring-context-3.0.xsd*

*http://www.springframework.org/schema/mvc*

*http://www.springframework.org/schema/mvc/spring-mvc-3.0.xsd"*>

<bean class=*"org.apache.solr.client.solrj.impl.HttpSolrServer"*>

<constructor-arg index=*"0"* value=*"${solr.server.baseUrl}"*></constructor-arg>

<property name=*"parser"*>

<bean class=*"org.apache.solr.client.solrj.impl.XMLResponseParser"*/>

</property>

<property name=*"maxRetries"* value=*"${solr.maxRetries}"*></property>

<property name=*"connectionTimeout"* value=*"${solr.connectionTimeout}"*></property>

</bean>

</beans>

实际上分词的效果并不够好，因为使用的分词器对 **连续英文数字分词** 的效果并不是很好，待解决。