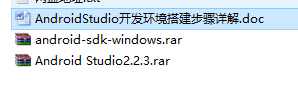
# 拷贝资料

链接：https://pan.baidu.com/s/16-T8VRBtw9E52kQTB0xTcQ

密码：3gxq

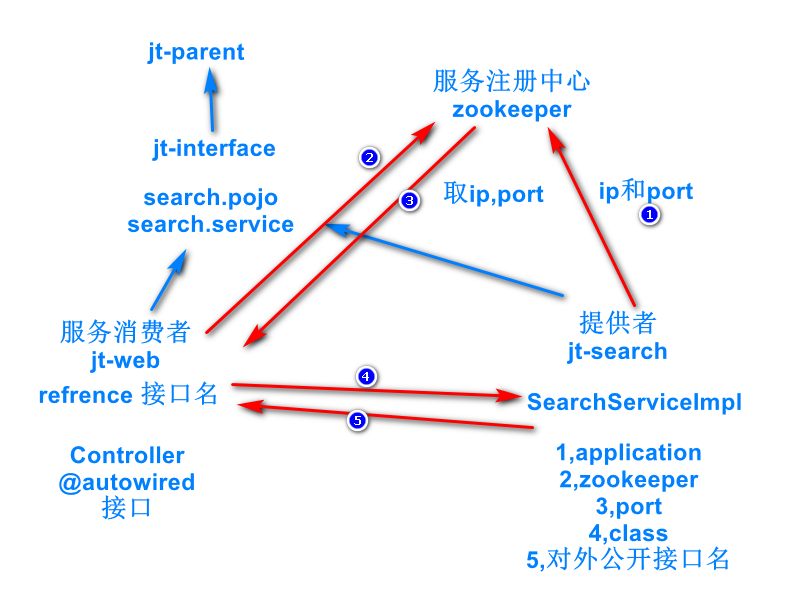
从别的同学拷贝网盘中的三个rar和一个文档



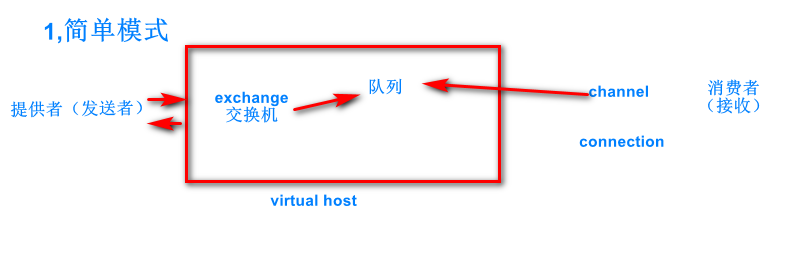
京淘项目每日课前资料day15-18.rar

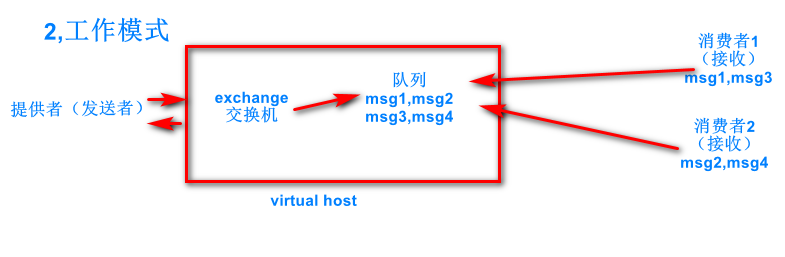
# 复习

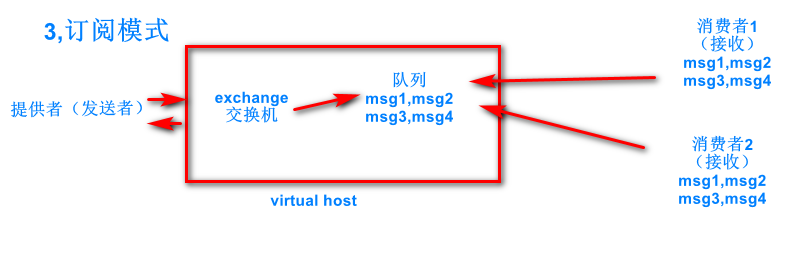
## Dubbo

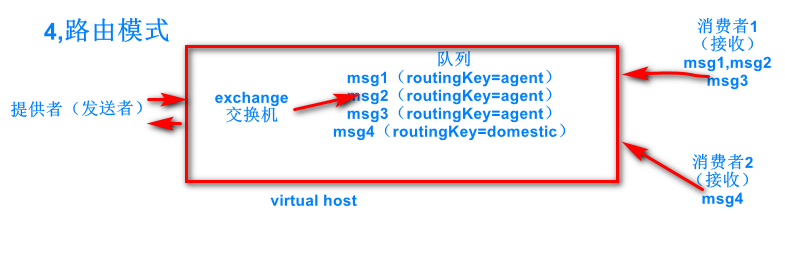


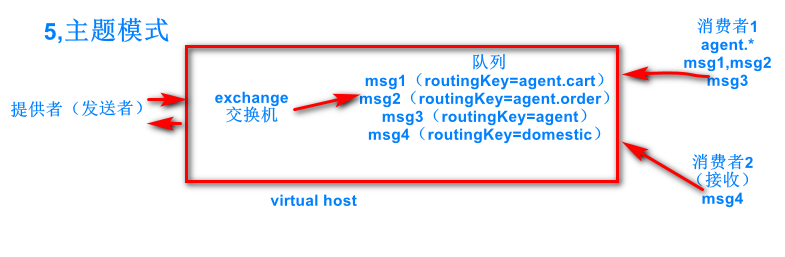
## 消息队列



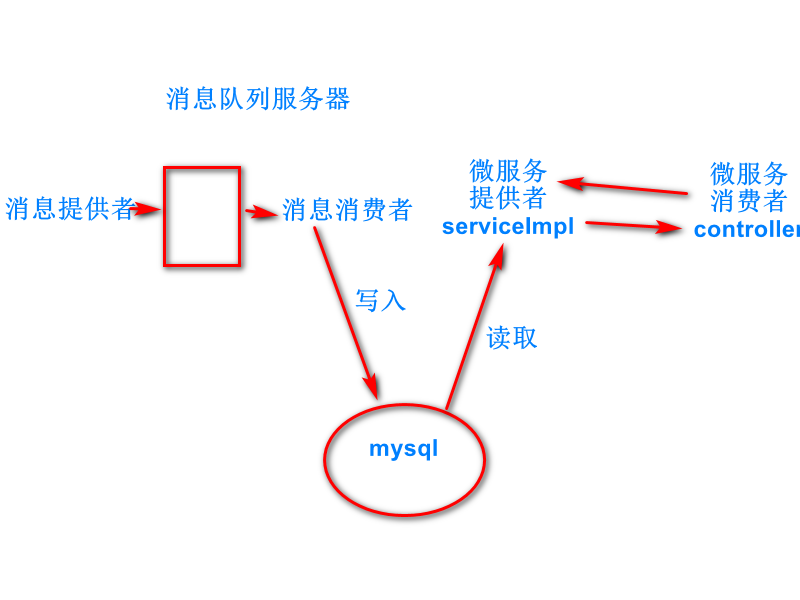








## Dubbo和消息队列一块使用



# 检索

## 原理



## Lucene

### 添加数据

**public** **class** LuceneTest {

//junit 版本是4.12

@Test

**public** **void** createIndex() **throws** Exception

{

//1,创建一个保存数据的文件夹

Directory directory=

FSDirectory.*open*(**new** File("./index"));

//2,指定一个中文分词器

Analyzer analyzer=**new** IKAnalyzer();

//3,设置配置信息

IndexWriterConfig config=

**new** IndexWriterConfig

(Version.***LUCENE\_4\_10\_2***, analyzer);

//4,创建一个写对象

IndexWriter indexWriter=

**new** IndexWriter(directory, config);

//5,创建要写的数据,document

Document document=**new** Document();

//6，设置数据的属性

//id,title,sellPoint.price

document.add(**new** LongField("id", 1, Store.***YES***));

document.add(**new**

StringField("title", "rose", Store.***YES***));

document.add(**new** TextField

("sellPoint","i love you",Store.***YES***));

document.add(**new** LongField

("price", 999900, Store.***YES***));

//7,写操作

indexWriter.addDocument(document);

// 8，关闭资源

indexWriter.close();

//运行，refresh项目，多出一个index文件夹

}

}

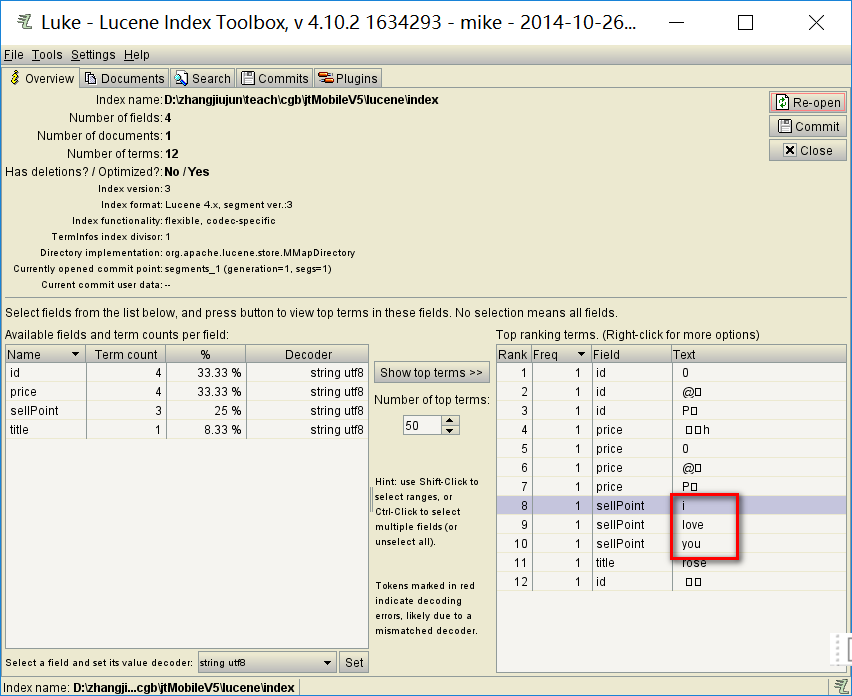
### 工具查看关键词(索引词)

\京淘项目每日课前资料\Day18-solr\查看索引工具\4.10.2\luke.bat

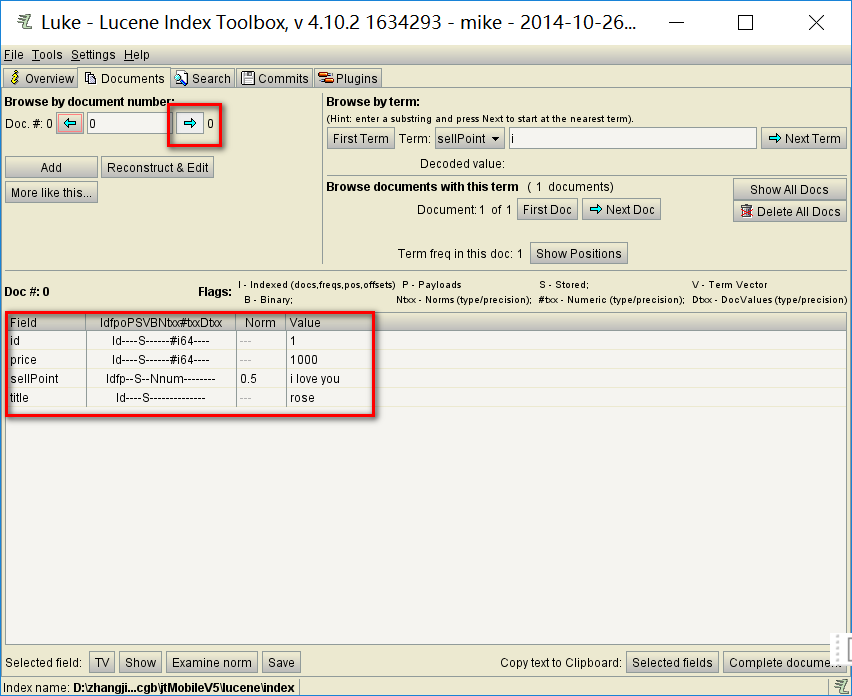
/09-jt/source/day15/查看索引工具/4.10.2

双击bat

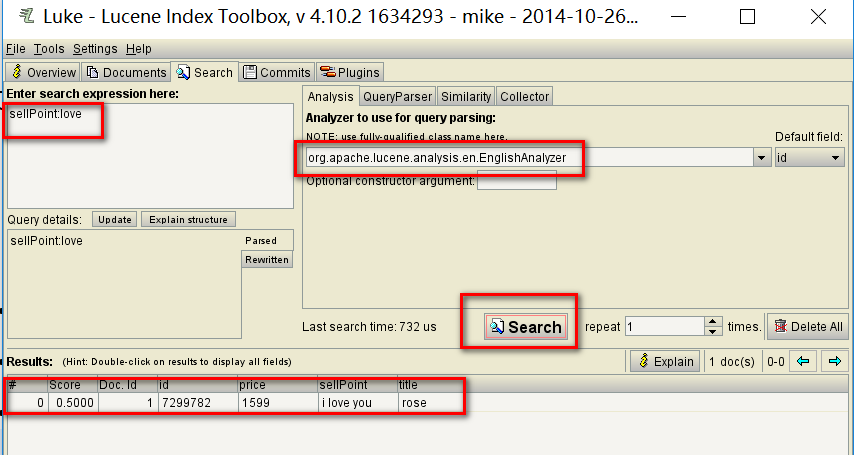
查看关键词



查看记录数



直接查询



### 代码进行查询

没有拿到网盘资料的同学下载

/09-jt/source/day18

**public** **class** SearchTest {

@Test

**public** **void** search() **throws** Exception

{

//1,指定从那查数据

Directory directory=

FSDirectory.*open*(**new** File("./index"));

//2,创建读的对象

IndexReader indexReader=

IndexReader.~~open~~(directory);

//3,创建查询对象

IndexSearcher indexSearcher=

**new** IndexSearcher(indexReader);

//4,指定关键字

Query query=**new** TermQuery

(**new** Term("sellPoint", "love"));

TopDocs topDocs=indexSearcher.search

(query, 20);

//5,遍历

**for**(ScoreDoc scoreDoc:topDocs.scoreDocs)

{

**int** index=scoreDoc.doc;

Document document=indexSearcher.doc(index);

String sellPoint=document.get("sellPoint");

System.***out***.println(index+":"+sellPoint);

}

//6,关闭资源

indexReader.close();

}

}

### Lucene中存在问题

说明:

可以为商品记录生成索引文件.但是需要每条信息都进行遍历.并且将item对象转化为Document对象之后才能创建索引.

缺点:

1. 效率太低
2. 如果商品修改了,则需要同步索引文件.
3. 如果商品新增了,则需要重新遍历重新索引.
4. 重新生成索引文件的时间较长.

## Solr

### 说明

Solr是一个独立的[企业级搜索](https://baike.baidu.com/item/%E4%BC%81%E4%B8%9A%E7%BA%A7%E6%90%9C%E7%B4%A2" \t "_blank)应用服务器，它对外提供类似于Web-service的API接口。用户可以通过http请求，向搜索引擎服务器提交一定格式的XML文件，生成索引；也可以通过Http Get操作提出查找请求，并得到XML格式的返回结果.

基于Lucene的全文搜索服务器。同时对其进行了扩展，提供了比Lucene更为丰富的查询语言，同时实现了可配置、可扩展并对查询性能进行了优化，并且提供了一个完善的功能管理界面，是一款非常优秀的[全文搜索引擎](https://baike.baidu.com/item/%E5%85%A8%E6%96%87%E6%90%9C%E7%B4%A2%E5%BC%95%E6%93%8E)。

特点:

1. solr可以根据数据库表自动生成索引文件.
2. Solr可以动态的定期自动更新索引(对更新的数据进行索引的修改)

### 同类型产品

网址:https://www.elastic.co/cn/products/elasticsearch



### 安装solr

\京淘项目每日课前资料\Day18-solr\solrLinux安装包\ solr-5.2.1.tgz

/09-jt/source/day15/solr-5.2.1.tgz

Solr要求jdk是1.8

使用fz上传到/usr/local/src中

解压 tar –xzvf solr-5.2.1.tgz

cd bin

sh solr start

在windows电脑上用浏览器访问<http://ip：8983>

### Core介绍

Solr中的core相当于lucene中的directory.

是个数据的集合，用来放数据

### 创建出错



#### 创建文件夹

cd /usr/local/src/solr-5.2.1

ls

cd server

ls

cd solr

mkdir jt jt/conf jt/data

#### 拷贝solrconfig.xml

京淘项目每日课前资料\Day18-solr\solrLinux安装包\02-所需配置文件\jt\conf

/09-jt/source/day18/02-所需配置文件.rar

### 修改dih-config.xml

SELECT id,title,sell\_point,price,num,barcode,image,cid,`status`,created,updated FROM tb\_item 只执行一次，把mysql中的数据读出来写到solr服务器

SELECT id,title,sell\_point,price,num,barcode,image,cid,`status`,created,updated FROM tb\_item WHERE updated &gt; date\_add(str\_to\_date('${dih.last\_index\_time}','%Y-%m-%d %H:%i:%s'),interval 8 hour)

每隔8个小时执行一次，把mysql修改的数据，新添置的数据再写到solr,或更新

<dataConfig>

<dataSource type="JdbcDataSource"

driver="com.mysql.jdbc.Driver"

url="jdbc:mysql://192.168.216.1:3306/jtdb"

user="root"

password="root"

batchSize="100"

autoCommit="false" />

<document name="item">

<entity name="item" pk="id"

query="SELECT id,title,sell\_point,price,num,barcode,image,cid,`status`,created,updated FROM tb\_item"

deltaQuery="SELECT id,title,sell\_point,price,num,barcode,image,cid,`status`,created,updated FROM tb\_item WHERE updated &gt; date\_add(str\_to\_date('${dih.last\_index\_time}','%Y-%m-%d %H:%i:%s'),interval 8 hour)"

transformer="RegexTransformer">

</entity>

</document>

</dataConfig>

### 修改schema.xml

<field name="id" type="long" indexed="true" stored="true" required="true" multiValued="false" />

<field name="title" type="text\_ik" indexed="true" stored="true"/>

<field name="sellPoint" type="text\_ik" indexed="true" stored="true"/>

<field name="price" type="long" indexed="true" stored="true"/>

<field name="num" type="int" indexed="true" stored="true"/>

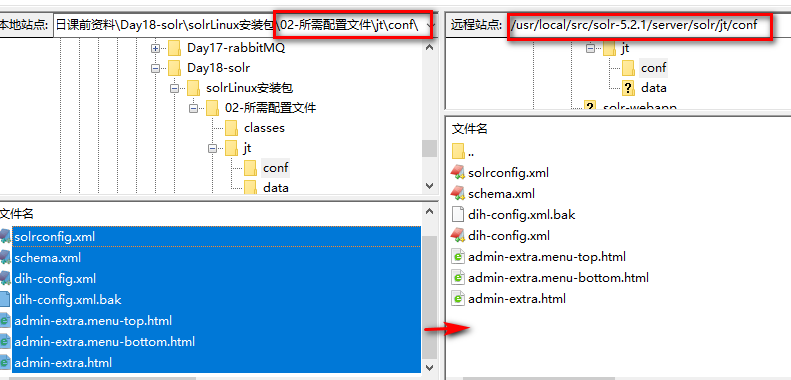
<field name="image" type="string" indexed="false" stored="true" multiValued="true"/>

<field name="created" type="date" indexed="true" stored="true"/>

<field name="updated" type="date" indexed="true" stored="true"/>

### 拷贝solr-config.xml等文件

拷贝到/usr/local/src/solr-5.2.1/server/solr/jt/conf

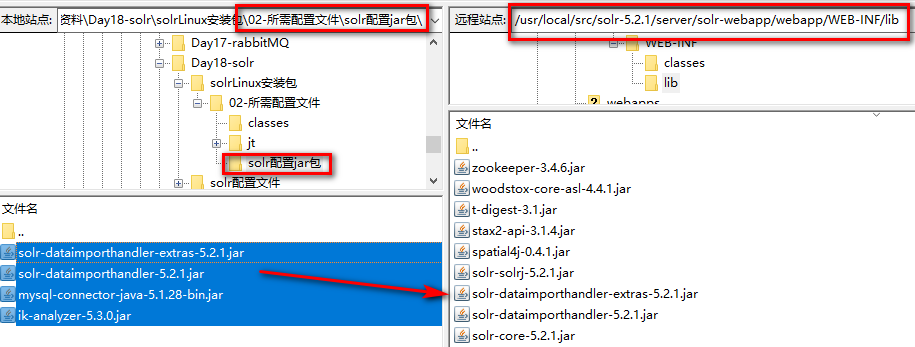


### 添加新的关键词

拷贝classes文件夹到

/usr/local/src/solr-5.2.1/server/solr-webapp/webapp/WEB-INF

### 拷贝mysql.jar



### 重启solr服务器

Cd /bin

Sh solr restart

cd /usr/local/src

[root@localhost src]# cd solr-5.2.1

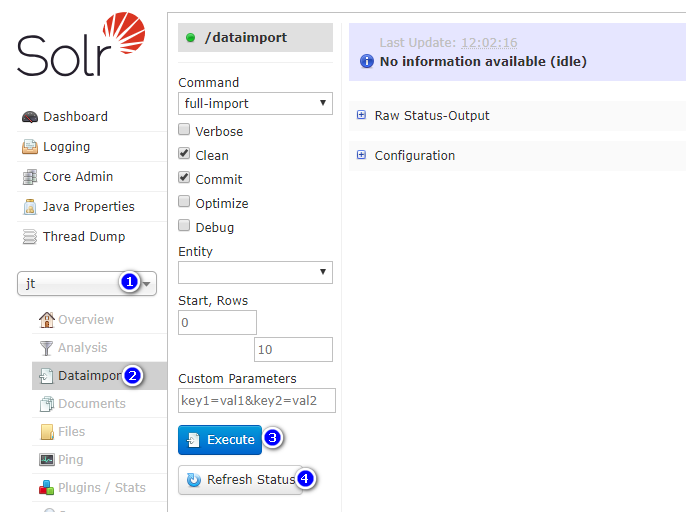
[root@localhost solr-5.2.1]# cd bin

[root@localhost bin]# pwd

/usr/local/src/solr-5.2.1/bin

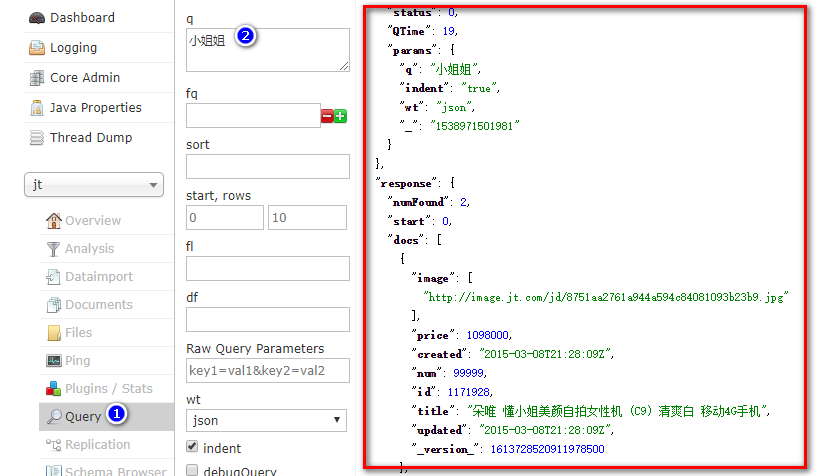
[root@localhost bin]# sh solr restart

### 导入数据





### 查询



没有下载网盘的长沙同学找以前班的项目经理要下面的资料

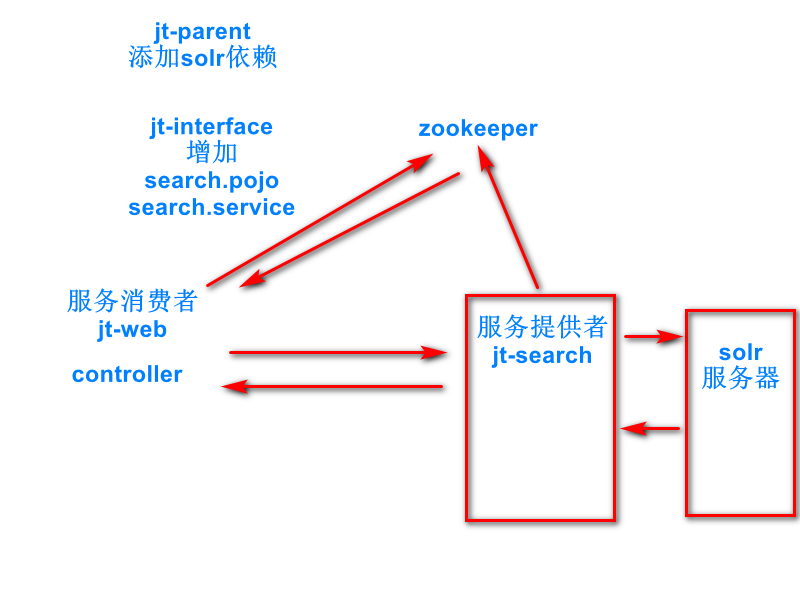


# 京淘项目中使用solr

## 需求

商品检索，搜索

## 分析



## 设计

## Jt-parent

添加solr依赖

<!-- 全文检索lucene -->

<dependency>

<groupId>org.apache.lucene</groupId>

<artifactId>lucene-core</artifactId>

<version>5.2.1</version>

</dependency>

<dependency>

<groupId>org.apache.lucene</groupId>

<artifactId>lucene-analyzers-common</artifactId>

<version>5.2.1</version>

</dependency>

<dependency>

<groupId>org.apache.lucene</groupId>

<artifactId>lucene-queryparser</artifactId>

<version>5.2.1</version>

</dependency>

<!-- IK分词器 -->

<dependency>

<groupId>org.wltea.analyzer</groupId>

<artifactId>ik-analyzer</artifactId>

<version>5.3.0</version>

</dependency>

<dependency>

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

<artifactId>solr-solrj</artifactId>

<version>5.2.1</version>

</dependency>

<!-- 爬虫jsoup -->

<dependency>

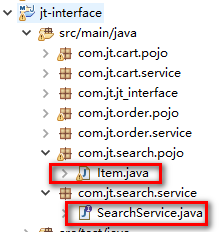
<groupId>org.jsoup</groupId>

<artifactId>jsoup</artifactId>

<version>1.9.1</version>

</dependency>

## Jt-interface增加pojo,service



//对应solr中的数据，solr中的数据来之mysql的tb\_item

**public** **class** Item **extends** BasePojo {

//id,title,price,sellPoint,image

//@Field设置属性id的值对应solr中数据的id field

@Field("id")

**private** Long id;

@Field("title")

**private** String title;

@Field("sellPoint")

**private** String sellPoint;

//controller把list<item>放到request中,jsp 中用jstl,foreach search.jsp中item.images

@Field("image")

**private** String images;

@Field("price")

**private** Long price;

**public** String[] getImages() {

//1.png,2.png

**return** images.split(",");

}

**public** **void** setImages(String images) {

**this**.images = images;

}

**public** Long getId() {

**return** id;

}

**public** **void** setId(Long id) {

**this**.id = id;

}

**public** String getTitle() {

**return** title;

}

**public** **void** setTitle(String title) {

**this**.title = title;

}

**public** String getSellPoint() {

**return** sellPoint;

}

**public** **void** setSellPoint(String sellPoint) {

**this**.sellPoint = sellPoint;

}

**public** Long getPrice() {

**return** price;

}

**public** **void** setPrice(Long price) {

**this**.price = price;

}

}

**public** **interface** SearchService {

**public** List<Item> findItemByKey(String key);

}

## Jt-search实现微服务

### 启动zookeeper,执行sh zkCli.sh

[root@localhost bin]# cd /usr/local/src

[root@localhost src]# cd zookeeper-3.4.8

[root@localhost zookeeper-3.4.8]# cd bin

[root@localhost bin]# ls

[root@localhost bin]# sh zkServer.sh start

[root@localhost bin]# sh zkCli.sh

[zk: localhost:2181(CONNECTED) 0] ls /

[dubbo, zookeeper]

[zk: localhost:2181(CONNECTED) 1] ls /dubbo

[com.jt.cart.service.CartService, com.tedu.service.CartService, com.jt.order.service.OrderService]

### 继承parent,依赖common,interface,dubbo

### 拷贝<build> 修改port

从jt-cart拷贝，修改端口号

<build>

<plugins>

<plugin>

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

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

<version>2.2</version>

<configuration>

<port>8099</port>

<path>/</path>

</configuration>

</plugin>

</plugins>

</build>

### 从jt-cart拷贝applicationContext.xml

只保留

<context:component-scan base-package=*"com.jt"*/>

<bean id="propertyPlaceholder" class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">

<property name="locations">

<list>

<!-- <value>classpath:/property/solr.properties</value> -->

</list>

</property>

</bean>

### 从jt-cart拷贝web.xml中的内容

从jt-cart中拷贝下面的内容

<servlet>

<servlet-name>springmvc</servlet-name>

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

<init-param>

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

<!--扫描全部以applicationContext开通的配置文件 -->

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

</init-param>

<!--让servlet第一个加载 -->

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

</servlet>

<servlet-mapping>

<servlet-name>springmvc</servlet-name>

<url-pattern>/</url-pattern>

</servlet-mapping>

<!--定义POST乱码解决 -->

<filter>

<filter-name>characterEncoding</filter-name>

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

<!--定义初始化参数 -->

<init-param>

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

<param-value>UTF-8</param-value>

</init-param>

</filter>

<filter-mapping>

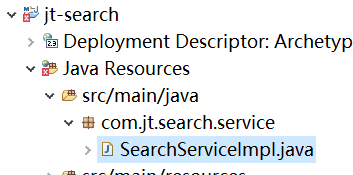
<filter-name>characterEncoding</filter-name>

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

</filter-mapping>

### 写实现类

添加searchServiceImpl



//微服务提供者

**public** **class** SearchServiceImpl

**implements** SearchService {

**public** List<Item> findItemByKey(String key) {

**return** **null**;

}

}

### 从jt-cart拷贝application-provider.xml

<dubbo:application name=*"provider-of-jtsearch"* />

<dubbo:registry address=*"zookeeper://192.168.216.200:2181"*>

</dubbo:registry>

<dubbo:protocol name=*"dubbo"* port=*"20885"*>

</dubbo:protocol>

<bean class=*"com.jt.search.service.SearchServiceImpl"* id=*"searchService"*>

</bean>

<dubbo:service interface=*"com.jt.search.service.SearchService"*

ref=*"searchService"*></dubbo:service>

### sh zki.sh

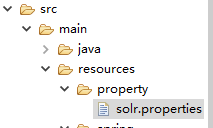
ls /dubbo

[com.jt.cart.service.CartService, com.jt.search.service.SearchService, com.tedu.service.CartService, com.jt.order.service.OrderService]

[zk: localhost:2181(CONNECTED) 3] ls /dubbo/com.jt.search.service.SearchService/providers

## Jt-search整合solr

### 添加properties文件



Solr.properties中的内容如下：

SOLR.URL=http://192.168.216.202:8983/solr/jt

### 在applicationContext中加载solr.properties

<bean id="propertyPlaceholder" class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">

<property name="locations">

<list>

<value>classpath:/property/solr.properties</value>

</list>

</property>

</bean>

### 添加solrJ配置文件

拷贝applicationContext.xml另存为applicationContext-solrj.xml

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

<constructor-arg index="0" value="${SOLR.URL}"/>

<!-- 设置响应解析器，solrj没有提供json解析器，所以通常用xml解析器 -->

<property name="parser">

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

</property>

<!-- 设置重试次数，推荐设置为1 -->

<property name="maxRetries" value="1"/>

<!-- 建立连接的最长时间 ,单位是：毫秒-->

<property name="connectionTimeout" value="500"/>

</bean>

### searchServiceImpl查询solr服务器

//微服务提供者

**public** **class** SearchServiceImpl

**implements** SearchService {

@Autowired

~~HttpSolrServer~~ httpSolrServer;

**public** List<Item> findItemByKey(String key) {

List<Item> itemList=**null**;

**try** {

SolrQuery solrQuery=**new** SolrQuery(key);

solrQuery.setStart(0);

solrQuery.setRows(20);

QueryResponse response=

httpSolrServer.query(solrQuery);

//item.class是类信息，通过类信息得到属性Field.getAnnotation()

itemList=response.getBeans(Item.**class**);

} **catch** (Exception e) {

// **TODO**: handle exception

}

**return** itemList;

}

}

## Jt-web调微服务

### 配置文件增加接口

applicationContext-consumer.xml

<dubbo:reference timeout=*"90000"* check=*"false"*

interface=*"com.jt.search.service.SearchService"*

id=*"searchService"*>

</dubbo:reference>

### Controller

@Controller

**public** **class** SearchController {

//由dubbo来提供searchService,

@Autowired

SearchService searchService;

@RequestMapping("/search")

**public** String search(String q,Model model)**throws** Exception

{

//chrome get 发信息的默认编码是iso-8859-1

**byte**[] data=q.getBytes("ISO-8859-1");

String key=**new** String(data, "UTF-8");

//dubbo消费者发信息给提供者，字符的编码是utf-8

//调用微服务的提供者

List<Item> itemList=searchService

.findItemByKey(key);

model.addAttribute("itemList", itemList);

**return** "search";

}

}

# 总结