**品优购电商系统开发**

**第10章**

**搜索解决方案-ElasticSearch**

传智播客.黑马程序员

# 课程目标

目标1：完成ElasticSearch安装与配置

目标2：掌握SpringDataElasticSearch入门

目标3：完成批量数据导入的功能

目标4：完成关键字搜索功能

# 1.ElasticSearch安装&配置

## 1.1商品搜索思考

大多数搜索引擎应用都必须具有某种搜索功能，问题是搜索功能往往是巨大的资源消耗并且它们由于沉重的数据库加载而拖垮你的应用的性能。

这就是为什么转移负载到一个外部的搜索服务器是一个不错的主意，ElasticSearch是一个流行的开源搜索服务。

ElasticSearch是一个开源高性能的搜索服务，适合用于构建高并发搜索应用程序。 它建立在[Lucene](http://www.yiibai.com/lucene/" \o "Lucene)(全文搜索引擎)之上。 ElasticSearch是企业级的，快速的和高度可扩展的。

## 1.2 ElasticSearch安装&配置

参考：资料\ElasticSearch相关资料\【**Linux安装ElasticSearch.docx**】

# 2.Spring Data ElasticSearch介绍&使用

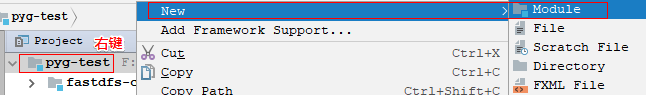
## 2.1 Spring Data ElasticSearch介绍

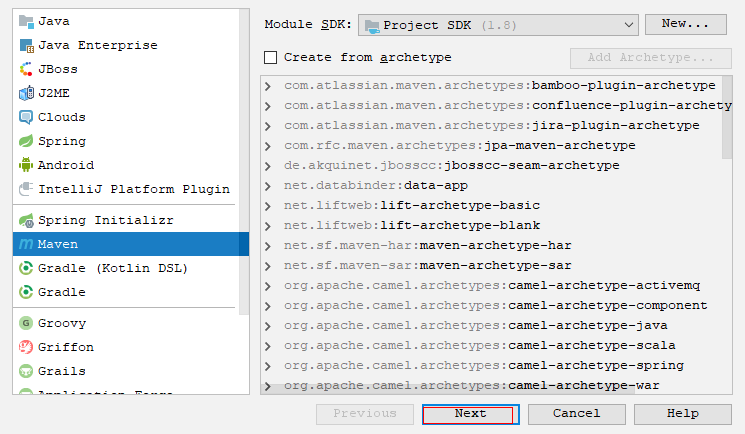
我们如何将ElasticSearch的应用集成到Spring应用中？Spring Data ElasticSearch就是为了简化ElasticSearch的java客户端开发，而研发的一个框架，其底层是对ElasticSearch的java客户端官方API的高度封装。

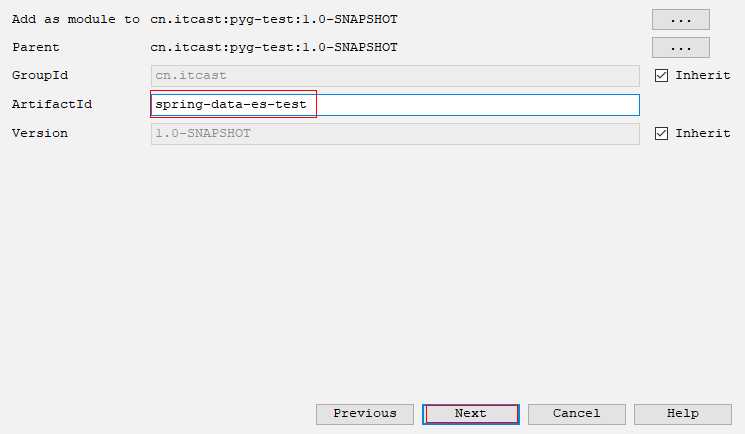
## 2.2 Spring Data ElasticSearch使用

### 2.2.1 搭建测试模块

spring-data-es-test模块(jar)







spring-data-es-test/pom.xml文件，配置依赖jar包

|  |
| --- |
| <**dependencies**>  ***<!-- 配置spring-data-elasticsearch -->* <dependency>  <groupId>org.springframework.data</groupId>  <artifactId>spring-data-elasticsearch</artifactId>  <version>3.1.3.RELEASE</version>  </dependency>**  *<!-- spring-test -->* <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-test</**artifactId**>  <**version**>5.1.3.RELEASE</**version**>  <**scope**>test</**scope**>  </**dependency**> </**dependencies**> |

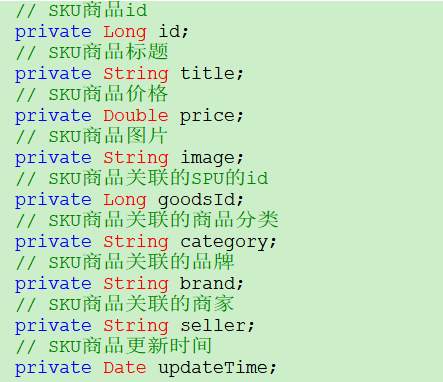
spring-data-es-test\src\main\resources\log4j2.xml日志文件

|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* <**Configuration status="warn"**>  <**Appenders**>  <**Console name="Console" target="SYSTEM\_OUT"**>  <**PatternLayout pattern="%m%n"**/>  </**Console**>  </**Appenders**>  <**Loggers**>  <**Root level="INFO"**>  <**AppenderRef ref="Console"**/>  </**Root**>  </**Loggers**> </**Configuration**> |

### 2.2.2 业务相关

* 索引(index): **pinyougou (pinyougou\_db数据库)**
* 类型(type): **item (tb\_item表)**
* 字段(field): **tb\_item表中的列(9列)**

(id、title、price、image、goodsId、category、brand、seller、updateTime)



### 2.2.3 注解介绍

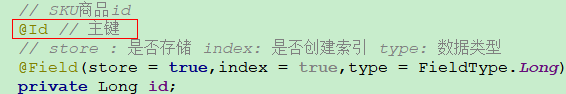
* **@Document 文档**: 把实体对象映射成一篇文档数据

-- indexName: 指定索引库的名称

-- type: 指定索引库中的类型(相当于表)



* **@Id 主键**: 对应数据库表中的主键列



* **@Field 字段**: 把数据库表中的列转化成文档中的Field

-- store: 是否存储 true存储|false不存储(**默认为false**)

-- index: 是否建立索引true建立|false不建立(**默认为true**)

-- type: 指定该Field的数据类型

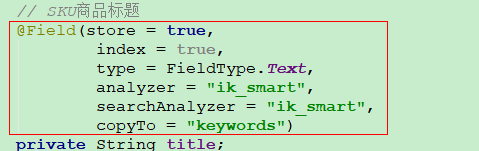
-- analyzer: 指定建立索引时的分词器

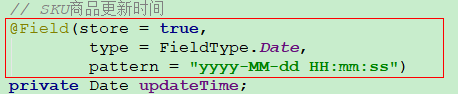
-- searchAnalyzer: 指定搜索时的分词器

-- copyTo: 复制到哪个Field(可以把多Field复制到一个Field，

这样做的好处搜索时方便指定搜索条件)

-- pattern: 日期类型，需要指定日期格式器





### 2.2.4 整合Spring Data ElasticSearch

#### **2.2.4.1实体类**

在spring-data-es-test/src/main/java/目录下创建 cn/itcast/es/pojo 包，将【资料\Item.java】拷入该包下面，加入Spring-Data-ElasticSearch的注解。

|  |
| --- |
| *// indexName: 索引库名称 type:类型名称(表)* **@Document**(indexName = **"pinyougou"**, type = **"item"**) **public class** Item **implements** Serializable {  *// SKU商品id* @Id *// 主键  // store : 是否存储 index: 是否创建索引 type: 数据类型* **@Field(store = true,index = true,type = FieldType.*Long*)**  **private** Long **id**;  *// SKU商品标题* **@Field(store = true,type = FieldType.*Text*,  analyzer = "ik\_smart",  searchAnalyzer = "ik\_smart")**  **private** String **title**;  *// SKU商品价格* **@Field(store = true,type = FieldType.*Double*)**  **private Double price**;  *// SKU商品图片* **@Field(store = true,index = false, type = FieldType.*Text*)**  **private** String **image**;  *// SKU商品关联的SPU的id* **@Field(store = true,type = FieldType.*Long*)**  **private** Long **goodsId**;  *// SKU商品关联的商品分类(不分词)* **@Field(store = true,type = FieldType.*Keyword*)**  **private** String **category**;  *// SKU商品关联的品牌(不分词)* **@Field(store = true,type = FieldType.*Keyword*)**  **private** String **brand**;  *// SKU商品关联的商家(不分词)* **@Field(store = true,type = FieldType.*Keyword*)**  **private** String **seller**;  *// SKU商品更新时间* **@Field(store = true, type = FieldType.*Date*, pattern = "yyyy-MM-dd HH:mm:ss")**  **private** Date **updateTime**;   */\*\* setter and getter method \*/*  *// 省略......* } |

#### **2.2.4.2配置部分**

spring-data-es-test/src/main/resources/applicationContext-elasticsearch.xml配置文件

|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* <**beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns:elasticsearch="http://www.springframework.org/schema/data/elasticsearch"  xsi:schemaLocation="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans.xsd  http://www.springframework.org/schema/data/elasticsearch  http://www.springframework.org/schema/data/elasticsearch/spring-elasticsearch.xsd"**>   *<!--* ***2.配置Elasticsearch客户端连接对象*** *cluster-nodes: 集群中的节点  cluster-name: 集群的名称  -->* <**elasticsearch:transport-client id="client"  cluster-nodes="192.168.12.131:9300"  cluster-name="elasticsearch"**/>   *<!--* ***1.配置ElasticsearchTemplate模板对象*** *-->* <**bean id="elasticsearchTemplate"  class="org.springframework.data.elasticsearch.core.ElasticsearchTemplate"**>  *<!-- 设置客户端连接对象 -->* <**constructor-arg name="client" ref="client"**/>  </**bean**>   *<!--* ***3.配置Elasticsearch数据访问接口，采用包扫描*** *-->* <**elasticsearch:repositories base-package="cn.itcast.es.dao"**/> </**beans**> |

#### **2.2.4.3索引数据访问接口**

spring-data-es-test\src\main\java\cn\itcast\es\dao\ItemDao.java

|  |
| --- |
| */\*\* Item数据访问接口 \*/* **public interface** ItemDao **extends ElasticsearchRepository<Item,Long**>{  } |

### 2.2.5 操作ElasticSearch

#### **2.2.5.1创建索引&映射**

spring-data-es-test\src\test\java\cn\itcast\es\SpringDataElasticSearchTest.java测试类

|  |
| --- |
| @RunWith(SpringJUnit4ClassRunner.**class**) @ContextConfiguration(locations = **"classpath:applicationContext-elasticsearch.xml"**) **public class** SpringDataElasticSearchTest {  @Autowired  **private** ElasticsearchTemplate **esTemplate**;  @Autowired(required = **false**)  **private** ItemDao **itemDao**;  *// 创建索引与映射* @Test  **public void** testCreateIndexAndMapping() {  *// 创建索引* **esTemplate**.createIndex(Item.**class**);  *// 创建映射* **esTemplate**.putMapping(Item.**class**);  } } |

#### **2.2.5.2添加或修改**

spring-data-es-test\src\test\java\cn\itcast\es\SpringDataElasticSearchTest.java测试类

|  |
| --- |
| *// 添加或修改类型(如果没有索引库与映射会自动创建)* @Test **public void** testSave(){  Item item = **new** Item();  item.setId(1L);  item.setTitle(**"华为Mate10"**);  item.setPrice(2020d);  item.setImage(**"1.jpg"**);  item.setGoodsId(1L);  item.setCategory(**"手机"**);  item.setBrand(**"华为"**);  item.setSeller(**"华为2号专卖店"**);  item.setUpdateTime(**new** Date());  *// 添加或修改类型* **itemDao**.save(item); } *// 批量添加或修改类型(如果没有索引库与映射会自动创建)* @Test **public void** testBatchSave(){  List<Item> itemList = **new** ArrayList<>();  **for** (**long** i = 1; i <= 1000; i++) {  Item item = **new** Item();  item.setId(i);  item.setTitle(**"华为Mate9"** + i);  item.setPrice(i \* 10);  item.setImage(**"1.jpg"**);  item.setGoodsId(1L);  item.setCategory(**"手机"**);  item.setBrand(**"华为"**);  item.setSeller(**"华为2号专卖店"**);  item.setUpdateTime(**new** Date());  itemList.add(item);  }  *// 批量添加或修改类型* **itemDao**.saveAll(itemList); } |

#### **2.2.5.3删除**

spring-data-es-test\src\test\java\cn\itcast\es\SpringDataElasticSearchTest.java测试类

|  |
| --- |
| */\*\* 根据主键id删除 \*/* @Test **public void** testDeleteById(){  **itemDao**.deleteById(1L); } */\*\* 根据条件删除 \*/* @Test **public void** testDeleteByItem(){  Item item = **new** Item();  item.setId(10L);  **itemDao**.delete(item); } */\*\* 删除全部 \*/* @Test **public void** testDeleteAll(){  **itemDao**.deleteAll(); } |

#### **2.2.5.4查询或搜索**

spring-data-es-test\src\test\java\cn\itcast\es\SpringDataElasticSearchTest.java测试类

|  |
| --- |
| */\*\* 根据主键id查询 \*/* @Test **public void** testFindById(){  Item item = **itemDao**.findById(1L).get();  System.***out***.println(item.getId() + **"\t"** + item.getTitle()); } */\*\* 查询全部 \*/* @Test **public void** testFindAll(){  Iterable<Item> items = **itemDao**.findAll();  **for** (Item item : items) {  System.***out***.println(item.getId() + **"\t"** + item.getTitle());  } } */\*\* 分页查询 \*/* @Test **public void** testFindByPage(){  *// 创建分页参数封装对象* Pageable pageable = PageRequest.*of*(0, 5);  *// 分页查询* Page<Item> page = **itemDao**.findAll(pageable);  System.***out***.println(**"总记录数: "** + page.getTotalElements());  System.***out***.println(**"总页数: "** + page.getTotalPages());  *// 获取分页数据* List<Item> items = page.getContent();  **for** (Item item : items) {  System.***out***.println(item.getId() + **"\t"** + item.getTitle());  } } */\*\* 通匹符搜索 \*/* @Test **public void** testWildcardQuery(){  *// 创建搜索查询对象 (wildcardQuery: 不分词)* SearchQuery query = **new** NativeSearchQuery(QueryBuilders  .*wildcardQuery*(**"title"**, **"华为"**));  *// 分页搜索，得到合计分页对象* AggregatedPage<Item> page = **esTemplate**.queryForPage(query, Item.**class**);  System.***out***.println(**"总记录数: "** + page.getTotalElements());  System.***out***.println(**"总页数: "** + page.getTotalPages());  *// 获取分页数据* List<Item> items = page.getContent();  **for** (Item item : items) {  System.***out***.println(item.getId() + **"\t"** + item.getTitle());  } } */\*\* 匹配分页搜索 \*/* @Test **public void** testMatchQuery(){  *// 创建搜索查询对象 (matchQuery: 分词)* SearchQuery query = **new** NativeSearchQuery(QueryBuilders.*matchQuery*(**"keywords"**, **"华为小米"**));  *// 创建分页参数封装对象* Pageable pageable = PageRequest.*of*(0, 5);  *// 设置分页* query.setPageable(pageable);  *// 分页搜索* AggregatedPage<Item> page = **esTemplate**.queryForPage(query, Item.**class**);  System.***out***.println(**"总记录数: "** + page.getTotalElements());  System.***out***.println(**"总页数: "** + page.getTotalPages());  *// 获取分页数据* List<Item> items = page.getContent();  **for** (Item item : items) {  System.***out***.println(item.getId() + **"\t"** + item.getTitle());  } } |

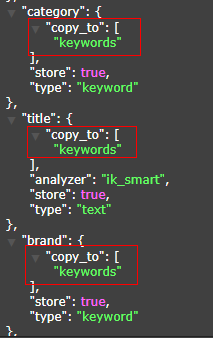
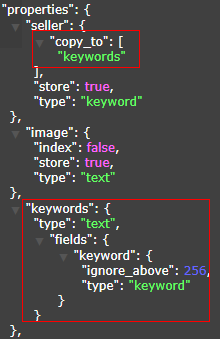
### 2.2.6复制Field

spring-data-es-test\src\main\java\cn\itcast\es\pojo\Item.java，增加**copyTo**属性，如下：



spring-data-es-test\src\test\java\cn\itcast\es\SpringDataElasticSearchTest.java测试类，运行testSave()方法后，查询索引库如下：





### 2.2.7嵌套Field

spring-data-es-test\src\main\java\cn\itcast\es\pojo\Item.java，用Map集合定义嵌套类型：

|  |
| --- |
| *// SKU商品规格选项(嵌套类型，多种规格)* @Field(store = **true**, type = FieldType.***Nested***) **private** Map<String,String> **spec**;  **public** Map<String, String> getSpec() {  **return spec**; } **public void** setSpec(Map<String, String> spec) {  **this**.**spec** = spec; } |

spring-data-es-test\src\test\java\cn\itcast\es\SpringDataElasticSearchTest.java测试类，修改testBatchSave()方法

|  |
| --- |
| *// 批量添加或修改类型(如果没有索引库与映射会自动创建)* @Test **public void** testBatchSave(){  List<Item> itemList = **new** ArrayList<>();  **for** (**long** i = 1; i <= 1000; i++) {  Item item = **new** Item();  item.setId(i);  item.setTitle(**"华为Mate9"** + i);  item.setPrice(i \* 10);  item.setImage(**"1.jpg"**);  item.setGoodsId(1L);  item.setCategory(**"手机"**);  item.setBrand(**"华为"**);  item.setSeller(**"华为2号专卖店"**);  item.setUpdateTime(**new** Date());   ***// 嵌套Field(规格选项)* Map<String, String> spec = new HashMap<>();  spec.put("网络","移动4G");  spec.put("机身内存","128G");  item.setSpec(spec);**    itemList.add(item);  }  *// 批量添加或修改类型* **itemDao**.saveAll(itemList); } |

嵌套Field最终效果(**多种规格**)：



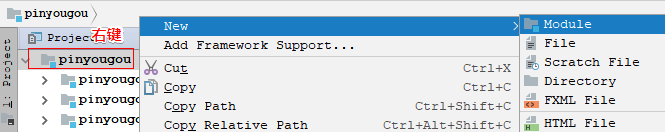
# 3.批量数据导入

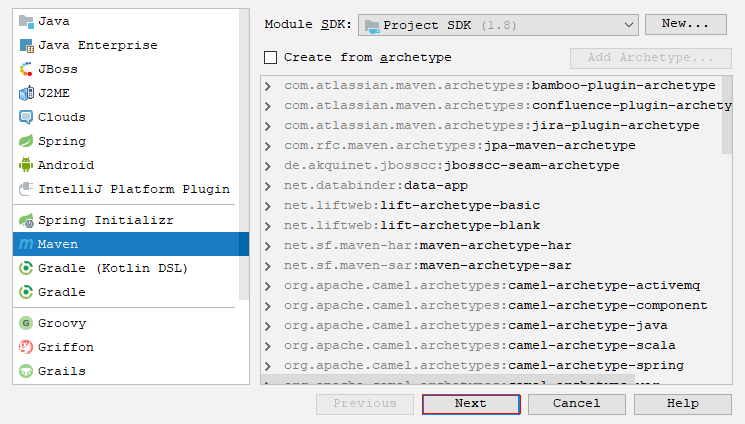
## 3.1需求分析

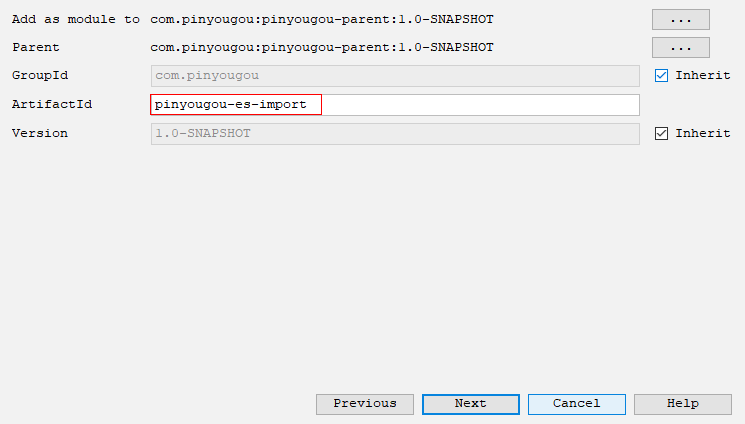
编写数据导入程序，将商品数据导入到ElasticSearch的pinyougou索引库中。

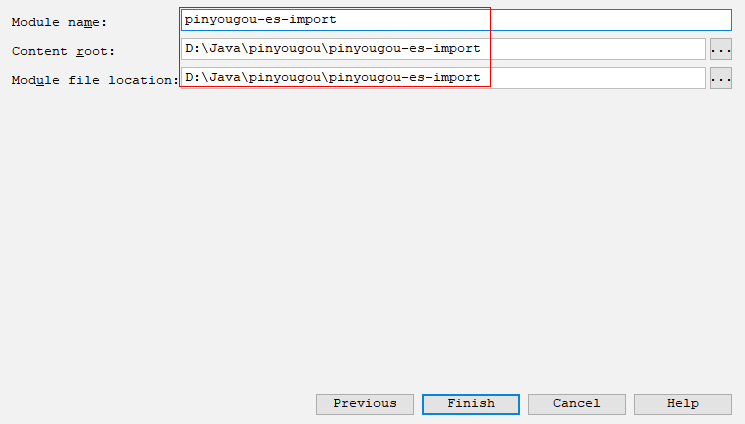
## 3.2搭建工程

创建pinyougou-es-import模块，打包方式jar：









pinyougou-es-import\pom.xml，引入依赖

|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* <**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"**>  <**parent**>  <**artifactId**>pinyougou-parent</**artifactId**>  <**groupId**>com.pinyougou</**groupId**>  <**version**>1.0-SNAPSHOT</**version**>  </**parent**>  <**modelVersion**>4.0.0</**modelVersion**>  <**artifactId**>pinyougou-es-import</**artifactId**>  <**dependencies**>  <**dependency**>  <**groupId**>com.pinyougou</**groupId**>  <**artifactId**>pinyougou-mapper</**artifactId**>  <**version**>1.0-SNAPSHOT</**version**>  </**dependency**>  *<!-- 配置spring-data-elasticsearch -->* <**dependency**>  <**groupId**>org.springframework.data</**groupId**>  <**artifactId**>spring-data-elasticsearch</**artifactId**>  </**dependency**>  </**dependencies**> </**project**> |

pinyougou-es-import\src\main\resources\applicationContext.xml

|  |
| --- |
| *<?***xml version="1.0" encoding="utf-8"***?>* <**beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns:context="http://www.springframework.org/schema/context"  xsi:schemaLocation="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans.xsd  http://www.springframework.org/schema/context  http://www.springframework.org/schema/context/spring-context.xsd"**>   *<!-- 配置加载属性文件 -->* <**context:property-placeholder  location="classpath\*:props/\*.properties"**/>  *<!-- 导入数据访问配置文件 -->* <**import resource="classpath:applicationContext-mapper.xml"**/>  *<!-- 配置组件扫描 -->* <**context:component-scan base-package="com.pinyougou.es"**/> </**beans**> |

## 3.3查询SKU商品数据

pinyougou-es-import\src\main\java\com\pinyougou\es\ItemImport.java，实现SKU商品数据的查询(上架的商品)：

|  |
| --- |
| */\*\* 数据导入 \*/* @Component **public class** ItemImport {  @Autowired  **private** ItemMapper **itemMapper**;   */\*\* 导入数据 \*/* **public void** importData(){  *// 创建Item对象封装查询条件* Item item = **new** Item();  *// 正常的商品* item.setStatus(**"1"**);  *// 从数据库表中查询SKU商品数据* List<Item> itemList = **itemMapper**.select(item);  System.***out***.println(**"===开始==="**);  **for**(Item item1 : itemList){  System.***out***.println(item1.getTitle());  }  System.***out***.println(**"===结束==="**);  }   **public static void** main(String[] args) {  ApplicationContext context = **new** ClassPathXmlApplicationContext(**"classpath:applicationContext.xml"**);  ItemImport itemImport = context.getBean(ItemImport.**class**);  itemImport.importData();  } } |

## 3.4集成SpringDataElasticSearch

### 3.4.1拷贝实体类

将spring-data-es-test测试模块中，添加了注解的实体类**Item.java**拷贝到pinyougou-pojo/src/main/java/com/pinyougou/es包下，重命名为**EsItem**。

### 3.4.2配置依赖

在pinyougou-pojo/pom.xml中，引入依赖

|  |
| --- |
| *<!-- 配置spring-data-elasticsearch -->* <**dependency**>  <**groupId**>org.springframework.data</**groupId**>  <**artifactId**>spring-data-elasticsearch</**artifactId**>  <**scope**>provided</**scope**> </**dependency**> |

### 3.4.3配置文件

修改pinyougou-es-import\src\main\resources\applicationContext.xml

|  |
| --- |
| *<?***xml version="1.0" encoding="utf-8"***?>* <**beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns:context="http://www.springframework.org/schema/context"  xmlns:elasticsearch="http://www.springframework.org/schema/data/elasticsearch"  xsi:schemaLocation="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans.xsd  http://www.springframework.org/schema/context  http://www.springframework.org/schema/context/spring-context.xsd  http://www.springframework.org/schema/data/elasticsearch  http://www.springframework.org/schema/data/elasticsearch/spring-elasticsearch.xsd"**>   *<!-- 配置加载属性文件 -->* <**context:property-placeholder  location="classpath\*:props/\*.properties"**/>  *<!-- 导入数据访问配置文件 -->* <**import resource="classpath:applicationContext-mapper.xml"**/>  *<!-- 配置组件扫描 -->* <**context:component-scan base-package="com.pinyougou.es"**/>    *<!-- ############ 集成Spring-Data-ElasticSearch ############ -->  <!-- 配置Elasticsearch客户端连接对象  cluster-nodes: 集群中的节点  cluster-name: 集群的名称  -->* <**elasticsearch:transport-client id="client"  cluster-nodes="192.168.12.131:9300"  cluster-name="elasticsearch"**/>  *<!-- 配置ElasticsearchTemplate模板对象 -->* <**bean id="elasticsearchTemplate"  class="org.springframework.data.elasticsearch.core.ElasticsearchTemplate"**>  *<!-- 设置客户端连接对象 -->* <**constructor-arg name="client" ref="client"**/>  </**bean**>  *<!-- 配置Elasticsearch数据访问接口，采用包扫描 -->* <**elasticsearch:repositories base-package="com.pinyougou.es.dao"**/> </**beans**> |

提供pinyougou-es-import\src\main\resources\log4j2.xml

|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* <**Configuration status="warn"**>  <**Appenders**>  <**Console name="Console" target="SYSTEM\_OUT"**>  <**PatternLayout pattern="%m%n"**/>  </**Console**>  </**Appenders**>  <**Loggers**>  <**Root level="INFO"**>  <**AppenderRef ref="Console"**/>  </**Root**>  </**Loggers**> </**Configuration**> |

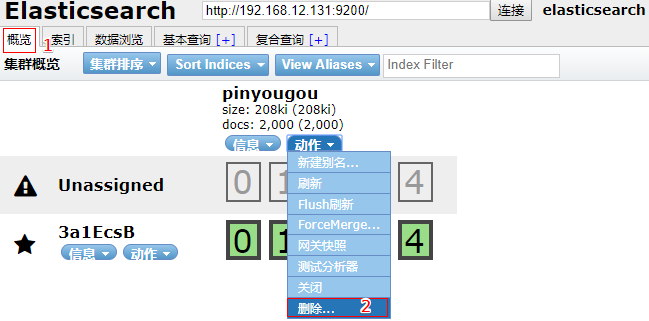
### 3.4.4索引数据访问接口

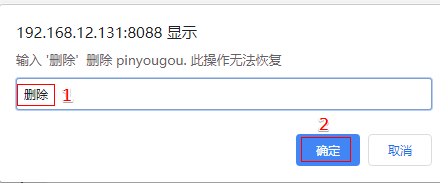
pinyougou-es-import\src\main\java\com\pinyougou\es\dao\EsItemDao.java，索引数据访问接口

|  |
| --- |
| **package** com.pinyougou.es.dao; **import** com.pinyougou.es.EsItem; **import** org.springframework.data.elasticsearch.repository.ElasticsearchRepository;  */\*\* EsItemDao数据访问接口 \*/* **public interface** EsItemDao **extends** ElasticsearchRepository<EsItem, Long>{ } |

## 3.5数据导入ElasticSearch索引库

在浏览器访问：<http://192.168.12.131:8088/>，删除pinyougou索引库



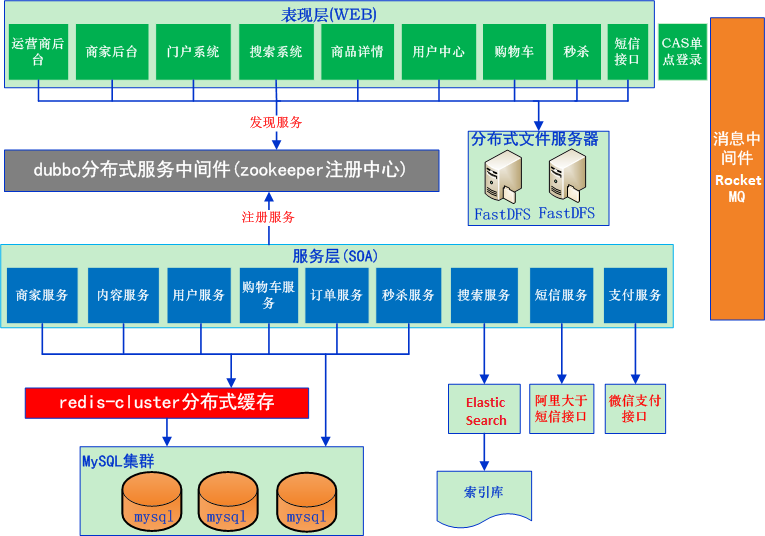


pinyougou-es-import\src\main\java\com\pinyougou\es\ItemImport.java，修改importData()方法：

|  |
| --- |
| */\*\* 数据导入 \*/* @Component **public class** ItemImport {  @Autowired  **private** ItemMapper **itemMapper**;  @Autowired  **private** EsItemDao **esItemDao**;   */\*\* 导入数据 \*/* **public void** importData() {  **try** {  *// 创建Item对象封装查询条件* Item item = **new** Item();  *// 上架的商品* item.setStatus(**"1"**);  *// 从数据库表中查询SKU商品数据* List<Item> itemList = **itemMapper**.select(item);   System.***out***.println(**"===开始==="**);  *// 定义EsItem集合* List<EsItem> esItems = **new** ArrayList<>();  **for** (Item item1 : itemList) {  EsItem esItem = **new** EsItem();  esItem.setId(item1.getId());  esItem.setTitle(item1.getTitle());  esItem.setPrice(item1.getPrice().doubleValue());  esItem.setImage(item1.getImage());  esItem.setGoodsId(item1.getGoodsId());  esItem.setCategory(item1.getCategory());  esItem.setBrand(item1.getBrand());  esItem.setSeller(item1.getSeller());  esItem.setUpdateTime(item1.getUpdateTime());   ***// 把SKU的规格json字符串，转化成Map集合(jackson)* Map<String, String> spec = new ObjectMapper()  .readValue(item1.getSpec(), Map.class);  *// 规格嵌套Field* esItem.setSpec(spec);**  *// 添加到集合* esItems.add(esItem);  }  *// 添加或修改索引库数据* **esItemDao**.saveAll(esItems);  System.***out***.println(**"===结束==="**);  }**catch** (Exception ex){  ex.printStackTrace();  }  }  **public static void** main(String[] args) {  ApplicationContext context = **new** ClassPathXmlApplicationContext(**"classpath:applicationContext.xml"**);  ItemImport itemImport = context.getBean(ItemImport.**class**);  itemImport.importData();  } } |



# 4.搭建搜索工程



## 4.1 pinyougou-search-service(war)

与pinyougou-content-service构建方式一样，拷贝pinyougou-content-service模块中的**web.xml、applicationContext-service.xml、log4j.properties**。

pinyougou-search-service/pom.xml：

|  |
| --- |
| <**properties**>  *<!-- 配置tomcat端口号 -->* **<port>9003</port>** </**properties**> <**dependencies**>  <**dependency**>  <**groupId**>org.springframework.data</**groupId**>  <**artifactId**>spring-data-elasticsearch</**artifactId**>  </**dependency**> </**dependencies**> |

pinyougou-search-service/src/main/resources/applicationContext-service.xml：

|  |
| --- |
| *<?***xml version="1.0" encoding="utf-8"***?>* <**beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns:dubbo="http://code.alibabatech.com/schema/dubbo"  xmlns:context="http://www.springframework.org/schema/context"  xsi:schemaLocation="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans.xsd  http://code.alibabatech.com/schema/dubbo  http://code.alibabatech.com/schema/dubbo/dubbo.xsd  http://www.springframework.org/schema/context  http://www.springframework.org/schema/context/spring-context.xsd"**>   *<!-- ############## 配置dubbo服务提供者 ############## -->  <!-- 配置当前应用的名称 -->* <**dubbo:application name="pinyougou-search-service"**/>  *<!-- 配置zookeeper作为注册中心，注册服务地址 -->* <**dubbo:registry protocol="zookeeper" address="192.168.12.131:2181"**/>  *<!-- 用dubbo协议在20882端口暴露服务 -->* <**dubbo:protocol name="dubbo" port="20882"**/>  *<!-- 配置采用包扫描来暴露服务 -->* <**dubbo:annotation package="com.pinyougou.search.service.impl"**/> </**beans**> |

提供pinyougou-search-service\src\main\resources\log4j2.xml，该文件为ElasticSearch的java客户端需要的日志文件。

|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* <**Configuration status="warn"**>  <**Appenders**>  <**Console name="Console" target="SYSTEM\_OUT"**>  <**PatternLayout pattern="%m%n"**/>  </**Console**>  </**Appenders**>  <**Loggers**>  <**Root level="INFO"**>  <**AppenderRef ref="Console"**/>  </**Root**>  </**Loggers**> </**Configuration**> |

## 4.2 pinyougou-search-web(war)

与pinyougou-portal-web构建方式一样，拷贝pinyougou-portal-web模块中的**web.xml、springmvc.xml、log4j.properties**

pinyougou-search-web/pom.xml：

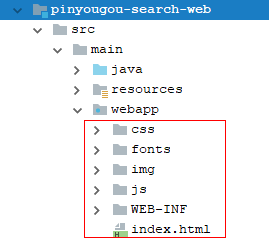
|  |
| --- |
| <**properties**>  *<!-- 覆盖父工程中的port属性 -->* <**port**>**9104**</**port**> </**properties**> |

pinyougou-search-web/src/main/resources/springmvc.xml：

|  |
| --- |
| *<?***xml version="1.0" encoding="utf-8"***?>* <**beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns:mvc="http://www.springframework.org/schema/mvc"  xmlns:dubbo="http://code.alibabatech.com/schema/dubbo"  xmlns:context="http://www.springframework.org/schema/context"  xsi:schemaLocation="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans.xsd  http://www.springframework.org/schema/mvc  http://www.springframework.org/schema/mvc/spring-mvc.xsd  http://code.alibabatech.com/schema/dubbo  http://code.alibabatech.com/schema/dubbo/dubbo.xsd  http://www.springframework.org/schema/context  http://www.springframework.org/schema/context/spring-context.xsd"**>   *<!-- 配置开启组件扫描 -->* **<context:component-scan base-package="com.pinyougou.search.controller"/>**   *<!-- 开启MVC注解驱动 -->* <**mvc:annotation-driven**>  *<!-- 配置消息转换 -->* <**mvc:message-converters**>  *<!-- 配置用fastjson作为json处理框架 -->* <**bean class="com.alibaba.fastjson.support.spring.FastJsonHttpMessageConverter"**>  *<!-- 配置支持的类型 -->* <**property name="supportedMediaTypes"  value="application/json;charset=UTF-8"**/>  </**bean**>  </**mvc:message-converters**>  </**mvc:annotation-driven**>  *<!-- 配置静态资源用WEB容器默认的servlet来处理 -->* <**mvc:default-servlet-handler**/>   *<!-- ############## 配置dubbo服务消费者 ############## -->  <!-- 配置当前应用的名称 -->* <**dubbo:application name="pinyougou-search-web"**/>  *<!-- 配置zookeeper作为注册中心，注册服务地址 -->* <**dubbo:registry protocol="zookeeper" address="192.168.12.131:2181"**/>  *<!-- 配置采用包扫描来引用服务，产生服务接口的代理对象 -->* <**dubbo:annotation package="com.pinyougou.search.controller"**/> </**beans**> |

## 4.3静态资源

拷贝【资料\搜索系统静态资源】至pinyougou-search-web/src/main/webapp/下面：



## 4.4配置域名访问

访问域名：<http://search.pinyougou.com>

配置hosts：127.0.0.1 search.pinyougou.com

配置Nginx反向代理：

|  |
| --- |
| server {  listen 80;  server\_name search.pinyougou.com;  proxy\_set\_header X-Forwarded-Host $host;  proxy\_set\_header X-Forwarded-Server $host;  location / {  proxy\_pass http://127.0.0.1:9104;  proxy\_connect\_timeout 600;  proxy\_read\_timeout 600;  }  } |

**注意：重启nginx (nginx -s reload)**

## 4.5集成SpringDataElasticSearch

pinyougou-search-service\src\main\resources\applicationContext-elasticsearch.xml：

|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* <**beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns:context="http://www.springframework.org/schema/context"  xmlns:elasticsearch="http://www.springframework.org/schema/data/elasticsearch"  xsi:schemaLocation="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans.xsd  http://www.springframework.org/schema/context  http://www.springframework.org/schema/context/spring-context.xsd  http://www.springframework.org/schema/data/elasticsearch  http://www.springframework.org/schema/data/elasticsearch/spring-elasticsearch.xsd"**>    *<!-- 配置加载属性文件 -->* <**context:property-placeholder  location="classpath\*:props/\*.properties"**/>   *<!-- 2. 配置ElasticSearch传输客户端对象 -->* <**elasticsearch:transport-client id="client"  cluster-nodes="${cluster-nodes}"  cluster-name="${cluster-name}"**/>   *<!-- 1. 配置ElasticsearchTemplate模板对象 -->* <**bean id="elasticsearchTemplate"  class="org.springframework.data.elasticsearch.core.ElasticsearchTemplate"**>  *<!-- 设置客户端连接对象 -->* <**constructor-arg name="client" ref="client"**/>  </**bean**>   *<!-- 3. 配置Elasticsearch数据访问接口，采用包扫描 -->* <**elasticsearch:repositories base-package="com.pinyougou.search.dao"**/> </**beans**> |

pinyougou-search-service\src\main\resources\**props**\application.properties：

|  |
| --- |
| *# 配置集群节点* **cluster-nodes**=**192.168.12.131:9300** *# 配置集群名称* **cluster-name**=**elasticsearch** |

pinyougou-search-service\src\main\java\com\pinyougou\search\dao\EsItemDao.java

|  |
| --- |
| */\*\* EsItem数据访问接口 \*/* **public interface** EsItemDao **extends ElasticsearchRepository<EsItem,Long**>{  } |

# 5.关键字搜索

## 5.1需求分析

打开搜索页面，在搜索框输入要搜索的关键字，点击搜索按钮即可进行搜索，显示搜索结果



## 5.2前端代码

pinyougou-search-web\src\main\webapp\index.html引入js

|  |
| --- |
| <**script src="/js/vue.min.js"**></**script**> <**script src="/js/axios.min.js"**></**script**> <**script src="/js/controller/searchController.js"**></**script**> |

index.html页面，绑定搜索框(76行)：

|  |
| --- |
| <**div class="input-append"**>  <**input type="text" id="autocomplete"  type="text" class="input-error input-xxlarge"  v-model="searchParam.keywords"**/>  <**button class="sui-btn btn-xlarge btn-danger"  @click="search()" type="button"**>搜索</**button**> </**div**> |

pinyougou-search-web/src/main/webapp/js/controller/searchController.js：

|  |
| --- |
| *// 窗口加载完* ***window***.onload = **function** () {  **var** vue = **new Vue**({  **el** : **'#app'**, *// 元素绑定* **data** : { *// 数据模型* **searchParam** : {**keywords** : **''**}, *// 搜索参数* **resultMap** : {} *// 搜索结果* },  **methods** : { *// 操作方法  // 商品搜索* search : **function** () {  *// 发送异步请求* **axios**.post(**"/search"**, **this**.**searchParam**)  .then(**function**(response){  *// 获取搜索结果: {}* vue.**resultMap** = response.**data**;  });  }  },  created : **function** () { *// 创建生命周期  // 调用商品搜索* **this**.search();  }  }); }; |

index.html页面，用v-for循环显示数据(287行)：

|  |
| --- |
| <**li class="yui3-u-1-5" v-for="item in resultMap.rows"**>  <**div class="list-wrap"**>  <**div class="p-img"**>  <**a href="item.html" target="\_blank"**><**img :src="item.image"**/></**a**>  </**div**>  <**div class="price"**>  <**strong**>  <**em**>¥</**em**>  <**i**>{{item.price}}</**i**>  </**strong**>  </**div**>  <**div class="attr"**>  <**em**>{{item.title}}</**em**>  </**div**>  <**div class="cu"**>  <**em**></**em**>  </**div**>  <**div class="commit"**>  <**i class="command"**>已有2000人评价</**i**>  </**div**>  <**div class="operate"**>  <**a href="success-cart.html" target="\_blank"  class="sui-btn btn-bordered btn-danger"**>加入购物车</**a**>  <**a href="javascript:void(0);" class="sui-btn btn-bordered"**>对比</**a**>  <**a href="javascript:void(0);" class="sui-btn btn-bordered"**>关注</**a**>  </**div**>  </**div**> </**li**> |

## 5.3后端代码

pinyougou-search-web/src/main/java/com/pinyougou/search/controller /包下，创建ItemSearchController.java：

|  |
| --- |
| @RestController **public class** ItemSearchController {  @Reference(timeout = **30000**)  **private** ItemSearchService **itemSearchService**;  @PostMapping(**"/search"**)  **public** Map<String, Object> search(@RequestBody  Map<String,Object> params){  **return itemSearchService**.search(params);  } } |

pinyougou-interface/src/main/java/com/pinyougou/service包下，创建服务接口ItemSearchService.java:

|  |
| --- |
| **public interface** ItemSearchService {  */\*\* 搜索方法 \*/* Map<String,Object> search(Map<String,Object> params); } |

pinyougou-search-service/src/main/java/com/pinyougou/search/service/impl包下，创建服务接口实现类ItemSearchServiceImpl.java：

|  |
| --- |
| */\*\* 商品搜索服务接口实现类 \*/* @Service(interfaceName = **"com.pinyougou.service.ItemSearchService"**) **public class** ItemSearchServiceImpl **implements** ItemSearchService {  @Autowired  **private** ElasticsearchTemplate **esTemplate**;  @Autowired  **private** EsItemDao **esItemDao**;   */\*\* 搜索方法 \*/* @Override  **public** Map<String, Object> search(Map<String,Object> params) {   */\*\* 获取查询关键字 \*/* String keywords = (String)params.get(**"keywords"**);   *// 创建原生的搜索条件构建对象* NativeSearchQueryBuilder builder = new NativeSearchQueryBuilder();  *// 设置默认搜索全部* builder.withQuery(QueryBuilders.*matchAllQuery*());  */\*\* 判断检索关键字是否为空 \*/* **if** (StringUtils.*isNoneBlank*(keywords)) {  *// 设置根据条件匹配查询: keywords 复制Field* builder.withQuery(QueryBuilders  .*matchQuery*("keywords", keywords));  }   *// 构建搜索查询对象* SearchQuery query = builder.build();  *// 分页查询，得到合计分页对象* AggregatedPage<EsItem> page = **esTemplate** .queryForPage(query, EsItem.**class**);   *// 定义Map集合封装返回数据* Map<String,Object> data = **new** HashMap<>();  *// 设置总记录数* data.put(**"total"**, page.getTotalElements());  *// 设置分页结果* data.put(**"rows"**, page.getContent());  **return** data;  } } |

# 6.Nginx+Tomcat【动静分离】

**问题：商品图片访问不了**

<http://image.pinyougou.com/jd/wKgMg1qzyiiAIACHAAN_6S46sOo724.jpg>

Nginx静态资源服务器，项目中所有的静态资源(css、js、images、html)，不需要经过tomcat服务器处理，用是用Nginx直接进行正向处理：

拷贝【资料\商品图片.zip】到F:盘并解压，效果如下：



访问域名：**[http://image.pinyougou.com](http://search.pinyougou.com)**

配置hosts：127.0.0.1 image.pinyougou.com

配置Nginx反向代理：

|  |
| --- |
| server {  listen 80;  server\_name image.pinyougou.com;  proxy\_set\_header X-Forwarded-Host $host;  proxy\_set\_header X-Forwarded-Server $host;  proxy\_set\_header Host $host;  location / {  root F:/pyg;  }  } |

**注意：需要重启Nginx反向代理服务器。**