# 今日任务

1. 熟悉企业数据迁移方式
2. 使用logstash迁移数据
3. 理解go8-mall搜索系统
4. 完成商品搜索服务及应用

# 问题？

**如何将其他关系型数据库里的数据导入到“es”中，以便用户检索数据？？**

可以自己写代码实现，难度不大，但是**通用性**不好，换个数据库，得修改代码，不灵活，容易有bug！！主要是还需要时间！！！

前面我给了个解决问题的原则，一般先是找开源的软件，如果没有再寻找商业的，或者企业自研。Java以开源闻名，发展了这么多年，只有你想不到的，没有你找不到的工具。如果没找到，就再找一次。

国内有阿里开源的**canal** ，但只局限于mysql数据库，实时性好，知名度也仅限于国内。

国际知名的开源的、专业的数据收集工具：**logstash**，这两年随着ES的崛起，越来越多的人，开始接触这个工具，该工具起初只是用来做日志收集的，用于著名的**ELK**日志框架中，但是由于其通用的、易扩展的、强大的数据收集能力，现在被越来越多的用在了数据库的数据迁移中。

# Logstash

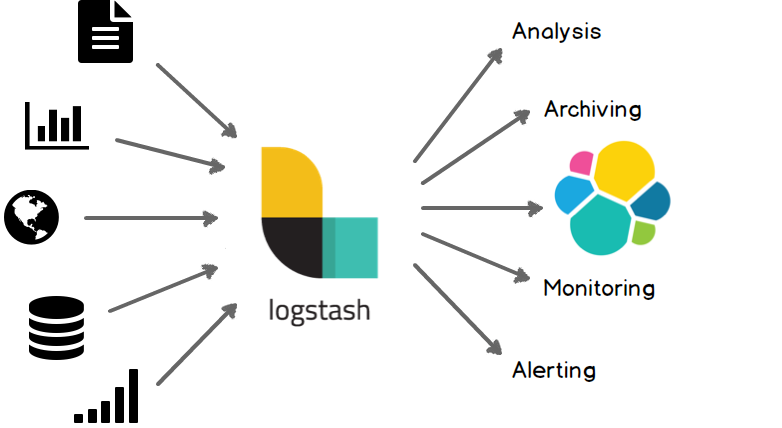
## 概念

Logstash 是开源的服务器端数据处理管道，能够同时从多个来源采集数据、转换数据，然后将数据发送到您最喜欢的 “存储库” 中。（我们的存储库当然是 Elasticsearch。）

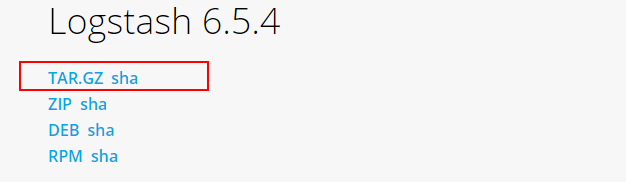
特点：

1. 作为es强大的数据收集主力
2. 灵活的插件式架构
3. 开发社区活跃，插件种类繁多

适用场合：



下载链接：<https://www.elastic.co/cn/downloads/past-releases/logstash-6-5-4>



## 安装

上传到虚拟机



入门配置，在根目录下新建一个配置文件



输入如下内容：

|  |
| --- |
| input { stdin { } }  output {  elasticsearch { hosts => ["192.168.192.50:9200"] }  stdout { codec => rubydebug }  } |

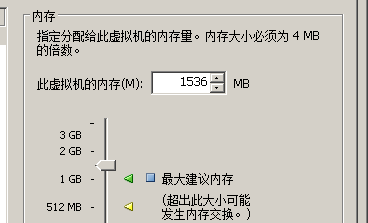
## 启动

bin/logstash -f logstash-simple.conf

**内存不够用**

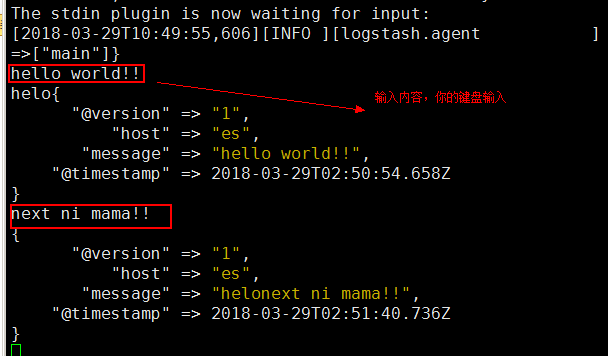


解决办法：增加虚拟机内存。可以不用停止虚拟机，减少内存不行。我们这里增加500M



启动后等待一段时间：

看到如下画面，

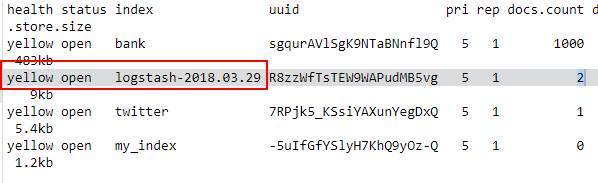


以上配置意思：从键盘输入，然后输出到屏幕和“es”中！！

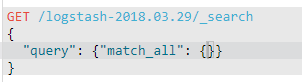
使用“**CTRL-D**”退出！！！

Es的结果：

**get /\_cat/indices?v**



查询：



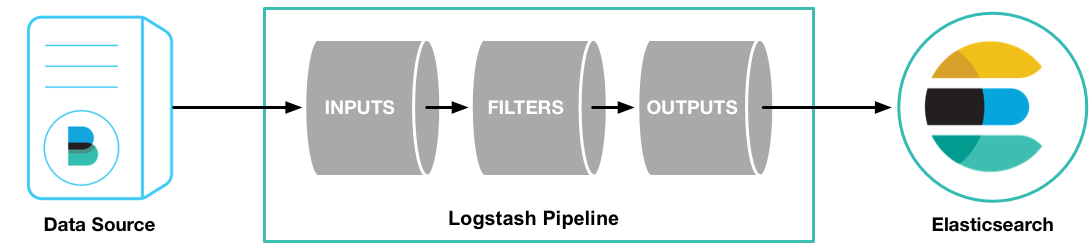


# 实战准备

## 原理

在深入使用logstash之前，先来理解logstash的流水线工作原理

流水线图：



最基本的流水线测试如下：

cd logstash-6.2.4

bin/logstash -e 'input { **stdin** { } } output { **stdout** {} }'

输出结果：

hello world

2013-11-21T01:22:14.405+0000 0.0.0.0 hello world

使用“**CTRL-D**”退出！！！

**流水线被抽象成三部分：输入、过滤以及输出，具体工作是由对应的插件来完成，插件式编程思维，在java的世界里很常见，Eclipse就是一个代表。**

## 目标

将目前存放在mysql中的商品信息导入到es中，以便于用户可以全文检索我们的商品！！

## 插件

首先要找到我们需要的插件：**logstash-input-jdbc**，并熟悉常用的配置。

1）需要手动在配置文件中指定相关的数据库的**驱动包**的路径！！配置选项：**jdbc\_driver\_library**

2） 定时任务配置采用的是cron风格

|  |
| --- |
| crontab文件的格式：M H D m d cmd.  M: 分钟（0-59）。  H：小时（0-23）。  D：天（1-31）。  m: 月（1-12）。  d: 一星期内的天（0~7，0和7均为星期天）。  cmd: 要执行的命令。 |

例如：

\* 5 \* 1-3 \* 1月到3月每天的早上5点的每一分钟执行一次

0 \* \* \* \* 每天每小时第0分钟执行

0 6 \* \* \* America/Chicago 每天早上6点(UTC/GMT -5)执行

3） 配置样例

|  |
| --- |
| input {  jdbc {  jdbc\_driver\_library => "mysql-connector-java-5.1.36-bin.jar"  jdbc\_driver\_class => "com.mysql.jdbc.Driver"  jdbc\_connection\_string => "jdbc:mysql://localhost:3306/mydb"  jdbc\_user => "mysql"  parameters => { "favorite\_artist" => "Beethoven" }  schedule => "\* \* \* \* \*"  statement => "SELECT \* from songs where artist = :favorite\_artist"  }  } |

Sql语句可以单独放在一个文件中，便于管理维护。

4） sql语句配置文件，在配置文件中参数名：**statement\_filepath**

默认一个插件里只能配置一条sql语句，如果配置多条它也只能执行一条。如果想执行多条sql语句，请配置多个插件即可！！

**注意点：**

当配置多条sql语句时，多个插件里需要配置各自的“**sql\_last\_value** ”，可以通过参数“**last\_run\_metadata\_path**”来指定，千万别放在一个文件里，要分开存放该值！！

其次，需要输出到es的输出插件：**logstash-output-elasticsearch，该插件不需要特殊配置，默认的就可以。**

# 实战练习

## 检查插件



结果都在：





## 准备驱动



按任务在根目录下建立一个任务目录，这里是“商品”



上传文件至该目录下



## 配置文件

goods.conf

|  |
| --- |
| input {  jdbc {  jdbc\_driver\_library => "/usr/local/logstash-6.5.4/goods/mysql-connector-java-8.0.13.jar"  jdbc\_driver\_class => "com.mysql.cj.jdbc.Driver"  jdbc\_connection\_string => "jdbc:mysql://192.168.19.1:3306/go8db?characterEncoding=utf8"  jdbc\_user => "root"  jdbc\_password => "111111"  statement\_filepath => "/usr/local/logstash-6.5.4/goods/goods.sql"  last\_run\_metadata\_path => "/usr/local/logstash-6.5.4/goods/goods.last"  }  }  output {  elasticsearch {  #ESIP地址与端口  hosts => "192.168.19.131:9200"  #ES索引名称（自己定义的）  index => "**goods**"  #自定义类型，默认是 doc  document\_type => "\_doc"  #自增ID编号  document\_id => "**%{id}**"  }  } |

该插件支持动态参数配置，格式“**%{field}**”，其中“field” 就是事件内容中字段名称！！

如何写sql语句是根据你的业务来的，以淘宝为例，搜索的页面结果如下：

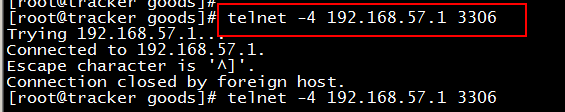


goods.sql

|  |
| --- |
| **select** p.id,p.title,c.name catalog\_name,b.cname brand\_cname,b.ename brand\_ename,p.price,p.attributes,p.picture **from** t\_product p **left** **join** t\_catalog c **on** p.cid=c.id **left** **join** t\_brand b **on** p.brand\_id=b.id **where** p.`status`=1 |

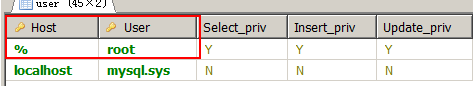
**小贴士：默认mysql不支持远程连接，请修改！！具体参考前面mysql教程！！**

**可以使用telnet工具先检测下mysql端口是否可以连接！！！！！**

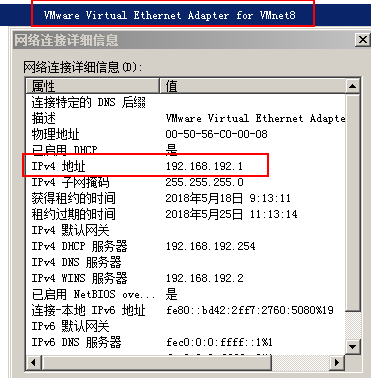
****

**添加远程连接权限**

|  |
| --- |
| **use** mysql;  **update** **user** **set** host='%' **where** **user**='root';  FLUSH PRIVILEGES |

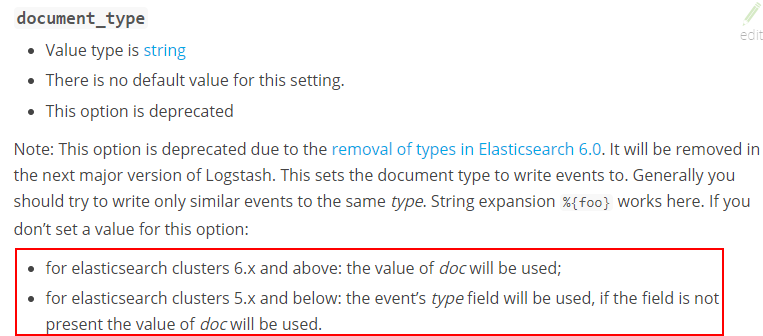


虚拟机（nat模式）访问本机的ip地址，参考：



## 创建索引

**请在导入数据之前先创建索引和映射，因为如果直接导入数据，ES默认创建的索引库文档是不能进行中文分词的。**



**创建索引时指定字段的分析器**

|  |
| --- |
| PUT goods  {  "mappings": {  "\_doc":{  "properties": {  "catalog\_name": {  "type": "keyword"  },  "brand\_cname": {  "type": "keyword"  },  "brand\_ename": {  "type": "keyword"  },  "id": {  "type": "long"  },  "price": {  "type": "integer"  },  "title": {  "type": "text",  "analyzer": "ik\_smart",  "fields": {  "keyword": {  "type": "keyword",  "ignore\_above": 256  }  }  },  "attributes": {  "type": "text",  "analyzer": "ik\_smart",  "fields": {  "keyword": {  "type": "keyword",  "ignore\_above": 256  }  }  },  "picture":{  "type": "text",  "index": false  }  }  }  }  } |

|  |
| --- |
| GET goods/\_search  {  "query": {"match\_all": {}}  } |

## 启动导入



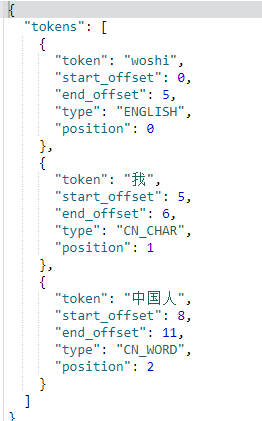
goods.last



## 验证中文分词



结果：



# 搜索系统

数据应用层

数 据 搜 索

数 据 处 理

数 据 来 源

logstash

数据库

文件系统

canal

消息系统

大数据

# 搜索服务

## 建立微服务工程

## Pom

|  |
| --- |
| <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.go9</groupId>  <artifactId>go9</artifactId>  <version>1.0.0-SNAPSHOT</version>  </parent>  <artifactId>go9-search</artifactId>  <properties>  <elasticsearch.version>6.5.4</elasticsearch.version>  </properties>  <dependencies>  <dependency>  <groupId>com.cjl</groupId>  <artifactId>elasticsearch-spring-boot-starter</artifactId>  <version>1.2.0.RELEASE</version>  </dependency>  <dependency>  <groupId>com.go9</groupId>  <artifactId>go9-common</artifactId>  <version>1.0.0-SNAPSHOT</version>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>  </dependency>  </dependencies>  </project> |

## Yml

|  |
| --- |
| eureka:  client:  serviceUrl:  defaultZone: http://localhost:8761/eureka/  server:  port: 8772  spring:  application:  name: sc-service-search  elasticsearch:  cluster-nodes: 192.168.192.140:9300  logging:  level:  root: info |

## 启动类

|  |
| --- |
| package com.sc.search;  import org.springframework.boot.SpringApplication;  import org.springframework.boot.autoconfigure.SpringBootApplication;  import org.springframework.cloud.netflix.eureka.EnableEurekaClient;  @SpringBootApplication  @EnableEurekaClient  public class SearchApplication {  public static void main(String[] args) {  SpringApplication.run(SearchApplication.class, args);  }  } |

## 分析业务

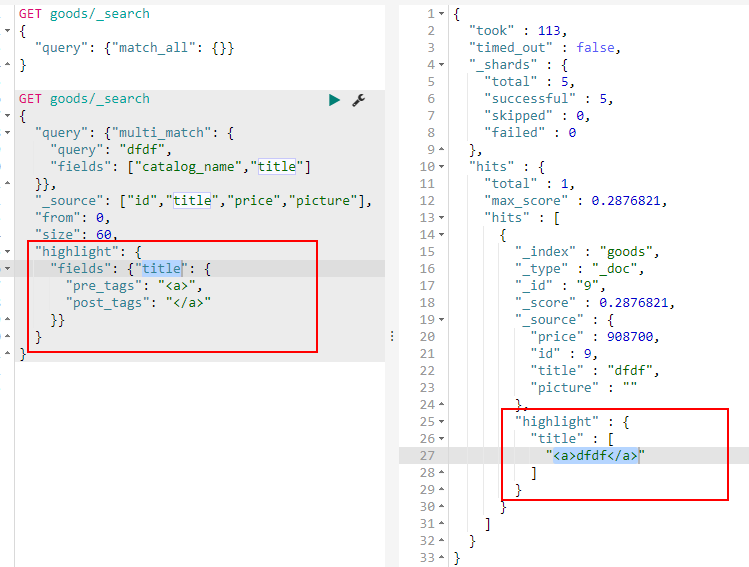
前端查询结果需要高亮显示，rest-api





搜索结果高亮，搜索引擎如何查询？？？

Rest-api接口查询风格就是这样的。



## 定义pojo

**Goods.java**

|  |
| --- |
| **package** com.go9.search.pojo;  **public** **class** Goods {  **private** String id;  **private** String brand\_ename;  **private** String attributes;  **private** **int** price;  **private** String title;  **private** String picture;  **private** String catalog\_name;  **private** String brand\_cname;    **public** String getId() {  **return** id;  }  **public** **void** setId(String id) {  **this**.id = id;  }  **public** String getBrand\_ename() {  **return** brand\_ename;  }  **public** **void** setBrand\_ename(String brand\_ename) {  **this**.brand\_ename = brand\_ename;  }  **public** String getAttributes() {  **return** attributes;  }  **public** **void** setAttributes(String attributes) {  **this**.attributes = attributes;  }  **public** **int** getPrice() {  **return** price;  }  **public** **void** setPrice(**int** price) {  **this**.price = price;  }  **public** String getTitle() {  **return** title;  }  **public** **void** setTitle(String title) {  **this**.title = title;  }  **public** String getPicture() {  **return** picture;  }  **public** **void** setPicture(String picture) {  **this**.picture = picture;  }  **public** String getCatalog\_name() {  **return** catalog\_name;  }  **public** **void** setCatalog\_name(String catalog\_name) {  **this**.catalog\_name = catalog\_name;  }  **public** String getBrand\_cname() {  **return** brand\_cname;  }  **public** **void** setBrand\_cname(String brand\_cname) {  **this**.brand\_cname = brand\_cname;  }    } |

SearchRecord.java

|  |
| --- |
| **package** com.go9.search.pojo;  **import** java.util.List;  **public** **class** SearchRecord<T> {  **private** **long** total;  **private** **int** page;  **private** **int** size;  **private** String keyword;  **private** List<T> records;    **public** String getKeyword() {  **return** keyword;  }  **public** **void** setKeyword(String keyword) {  **this**.keyword = keyword;  }  **public** **long** getTotal() {  **return** total;  }  **public** **void** setTotal(**long** total) {  **this**.total = total;  }  **public** **int** getPage() {  **return** page;  }  **public** **void** setPage(**int** page) {  **this**.page = page;  }  **public** **int** getSize() {  **return** size;  }  **public** **void** setSize(**int** size) {  **this**.size = size;  }  **public** List<T> getRecords() {  **return** records;  }  **public** **void** setRecords(List<T> records) {  **this**.records = records;  }  } |

## Dao

|  |
| --- |
| package com.go9.search.dao;  import java.util.ArrayList;  import java.util.List;  import java.util.Map;  import org.elasticsearch.action.search.SearchResponse;  import org.elasticsearch.client.Client;  import org.elasticsearch.common.text.Text;  import org.elasticsearch.index.query.QueryBuilders;  import org.elasticsearch.search.SearchHit;  import org.elasticsearch.search.fetch.subphase.highlight.HighlightBuilder;  import org.elasticsearch.search.fetch.subphase.highlight.HighlightField;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Repository;  import com.go9.search.pojo.Goods;  import com.go9.search.pojo.SearchRecord;  @Repository  public class GoodsDaoImpl implements GoodsDao {  @Autowired  private Client client;    @Override  public SearchRecord<Goods> search(String keyword,int page,int size) {  SearchRecord<Goods> sr = new SearchRecord<Goods>();    SearchResponse response = client.prepareSearch("goods")  .setQuery(QueryBuilders.multiMatchQuery(keyword, "title","catalog\_name"))  .setFrom((page-1)\*size).setSize(size)  .setFetchSource(new String[] {"id","title","price","picture"},null)  .highlighter(new HighlightBuilder().field("title")  .preTags("<span style=\"color:red\">")  .postTags("</span>"))  .execute().actionGet();    //底层解析  List<Goods> goodds = new ArrayList<>();  SearchHit[] hits = response.getHits().getHits();  for(SearchHit hit:hits) {  Goods goods = new Goods();  Map<String, Object> source = hit.getSourceAsMap();  goods.setId(source.get("id").toString());  goods.setPicture(source.get("picture").toString());  goods.setPrice(Integer.parseInt(source.get("price").toString()));    //高亮显示  Map<String, HighlightField> highlightFields = hit.getHighlightFields();  HighlightField highlightField = highlightFields.get("title");    if(highlightField!=null && highlightField.getFragments()!=null) {  Text[] fragments = highlightField.getFragments();  String title = fragments[0].toString();  goods.setTitle(title);  }else {  goods.setTitle(source.get("title").toString());  }  goodds.add(goods);  }  sr.setKeyword(keyword);  sr.setSize(size);  sr.setPage(page);  sr.setTotal(response.getHits().getTotalHits());  sr.setRecords(goodds);  return sr;  }  } |

## 业务层

|  |
| --- |
| **package** com.go9.search.service;  **import** org.springframework.beans.factory.annotation.Autowired;  **import** org.springframework.stereotype.Service;  **import** com.go9.search.dao.GoodsDao;  **import** com.go9.search.pojo.Goods;  **import** com.go9.search.pojo.SearchRecord;  @Service  **public** **class** GoodsServiceImpl **implements** GoodsService {  @Autowired  **private** GoodsDao goodsDao;    @Override  **public** SearchRecord<Goods> searchByKeyWord(String keyword, **int** page, **int** size) {  **return** goodsDao.search(keyword, page, size);  }  } |

## 封装返回对象

|  |
| --- |
| **package** com.go9.common;  **import** com.go9.common.ServiceMessage;  **public** **class** ServiceMessage<T> {  **private** **int** status;  **private** String message;  **private** T data;  **public** ServiceMessage() {  }  **public** ServiceMessage(**int** status, String message) {  **this**.status = status;  **this**.message = message;  }  **public** ServiceMessage(**int** status, String message, T data) {  **this**.status = status;  **this**.message = message;  **this**.data = data;  }  // 成功时status==200,message=ok  **public** **static** <T> ServiceMessage<T> ok() {  **return** **new** ServiceMessage<T>(200, "OK");  }  // 成功时status==200,message=ok  **public** **static** <T> ServiceMessage<T> ok(T data) {  **return** **new** ServiceMessage<T>(200, "OK", data);  }  // 成功时status==200,message=ok  **public** **static** ServiceMessage error() {  **return** **new** ServiceMessage(500, "ERROR");  }  **public** **int** getStatus() {  **return** status;  }  **public** **void** setStatus(**int** status) {  **this**.status = status;  }  **public** String getMessage() {  **return** message;  }  **public** **void** setMessage(String message) {  **this**.message = message;  }  **public** T getData() {  **return** data;  }  **public** **void** setData(T data) {  **this**.data = data;  }  } |

## 发布服务

|  |
| --- |
| **package** com.go9.search.controller;  **import** org.springframework.beans.factory.annotation.Autowired;  **import** org.springframework.web.bind.annotation.GetMapping;  **import** org.springframework.web.bind.annotation.RequestMapping;  **import** org.springframework.web.bind.annotation.RequestParam;  **import** org.springframework.web.bind.annotation.RestController;  **import** com.go9.common.ServiceMessage;  **import** com.go9.search.pojo.Goods;  **import** com.go9.search.pojo.SearchRecord;  **import** com.go9.search.service.GoodsService;  @RestController  @RequestMapping("/search")  **public** **class** GoodsController {  @Autowired  **private** GoodsService goodsService;    //发布一个接口  @GetMapping  **public** ServiceMessage<SearchRecord<Goods>> search(String keyword,@RequestParam(defaultValue="1")**int** page,@RequestParam(defaultValue="60")**int** size){  **try** {  SearchRecord<Goods> searchRecord = goodsService.searchByKeyWord(keyword, page, size);  **return** ServiceMessage.*ok*(searchRecord);  } **catch** (Exception e) {  e.printStackTrace();  **return** ServiceMessage.*error*();  }  }  } |

## 测试

先启动注册服务器，然后启动搜索服务。

# 搜索应用

在go8-mall门户工程中添加商品搜索页面，业务层调用商品搜索服务，获取搜索结果数据，然后在页面通过动态模板显示商品信息。

go8-portal

eureka-client

go8-search

eureka-client

## 业务层

|  |
| --- |
| **package** com.go9.mall.service;  **import** org.springframework.beans.factory.annotation.Autowired;  **import** org.springframework.beans.factory.annotation.Value;  **import** org.springframework.core.ParameterizedTypeReference;  **import** org.springframework.http.HttpEntity;  **import** org.springframework.http.HttpHeaders;  **import** org.springframework.http.HttpMethod;  **import** org.springframework.http.MediaType;  **import** org.springframework.http.ResponseEntity;  **import** org.springframework.stereotype.Service;  **import** org.springframework.web.client.RestTemplate;  **import** com.go9.common.ServiceMessage;  **import** com.go9.mall.pojo.Goods;  **import** com.go9.mall.pojo.SearchRecord;  @Service  **public** **class** GoodsServiceImpl **implements** GoodsService {  @Value("${search.url}")  **private** String searchUrl;    @Autowired  **private** RestTemplate template;    @Override  **public** SearchRecord<Goods> searchByKeyWord(String q, **int** size, **int** page) {  HttpHeaders headers = **new** HttpHeaders();  headers.add(HttpHeaders.***ACCEPT***, MediaType.***APPLICATION\_JSON\_UTF8\_VALUE***);  headers.add(HttpHeaders.***CONTENT\_TYPE***, MediaType.***APPLICATION\_JSON\_UTF8\_VALUE***);  HttpEntity requestEntity = **new** HttpEntity<>(headers);    ParameterizedTypeReference<ServiceMessage<SearchRecord<Goods>>> typeRef =  **new** ParameterizedTypeReference<ServiceMessage<SearchRecord<Goods>>>() {};  ResponseEntity<ServiceMessage<SearchRecord<Goods>>> exchange =  template.exchange(searchUrl+"?keyword="+q+"&page="+page+"&size="+size, HttpMethod.***GET***, requestEntity, typeRef);  ServiceMessage<SearchRecord<Goods>> body = exchange.getBody();  **if**(body.getStatus() == 200) {  SearchRecord<Goods> record = body.getData();  **return** record;  }**else** {  **return** **null**;  }  }  } |

## 发布层

|  |
| --- |
| **package** com.go9.mall.controller;  **import** org.springframework.beans.factory.annotation.Autowired;  **import** org.springframework.stereotype.Controller;  **import** org.springframework.ui.Model;  **import** org.springframework.web.bind.annotation.GetMapping;  **import** org.springframework.web.bind.annotation.RequestParam;  **import** com.go9.mall.pojo.Goods;  **import** com.go9.mall.pojo.SearchRecord;  **import** com.go9.mall.service.GoodsService;  @Controller  **public** **class** GoodsController {  @Autowired  **private** GoodsService goodsService;    @GetMapping("/list.html")  **public** String list(String q,@RequestParam(defaultValue="60")**int** size,@RequestParam(defaultValue="1")**int** page,Model model) {  SearchRecord<Goods> sr = goodsService.searchByKeyWord(q, size, page);  model.addAttribute("sr", sr);  **return** "list";  }  } |

## Pojo

|  |
| --- |
| **package** com.go9.mall.pojo;  **import** java.util.List;  **public** **class** SearchRecord<T> {  **private** **long** total;  **private** **int** page;  **private** **int** size;  **private** String keyword;  **private** List<T> records;    **public** String getKeyword() {  **return** keyword;  }  **public** **void** setKeyword(String keyword) {  **this**.keyword = keyword;  }  **public** **long** getTotal() {  **return** total;  }  **public** **void** setTotal(**long** total) {  **this**.total = total;  }  **public** **int** getPage() {  **return** page;  }  **public** **void** setPage(**int** page) {  **this**.page = page;  }  **public** **int** getSize() {  **return** size;  }  **public** **void** setSize(**int** size) {  **this**.size = size;  }  **public** List<T> getRecords() {  **return** records;  }  **public** **void** setRecords(List<T> records) {  **this**.records = records;  }  } |

|  |
| --- |
| **package** com.go9.mall.pojo;  **public** **class** Goods {  **private** String id;  **private** String brand\_ename;  **private** String attributes;  **private** **int** price;  **private** String title;  **private** String picture;  **private** String catalog\_name;  **private** String brand\_cname;    **public** String getId() {  **return** id;  }  **public** **void** setId(String id) {  **this**.id = id;  }  **public** String getBrand\_ename() {  **return** brand\_ename;  }  **public** **void** setBrand\_ename(String brand\_ename) {  **this**.brand\_ename = brand\_ename;  }  **public** String getAttributes() {  **return** attributes;  }  **public** **void** setAttributes(String attributes) {  **this**.attributes = attributes;  }  **public** **int** getPrice() {  **return** price;  }  **public** **void** setPrice(**int** price) {  **this**.price = price;  }  **public** String getTitle() {  **return** title;  }  **public** **void** setTitle(String title) {  **this**.title = title;  }  **public** String getPicture() {  **return** picture;  }  **public** **void** setPicture(String picture) {  **this**.picture = picture;  }  **public** String getCatalog\_name() {  **return** catalog\_name;  }  **public** **void** setCatalog\_name(String catalog\_name) {  **this**.catalog\_name = catalog\_name;  }  **public** String getBrand\_cname() {  **return** brand\_cname;  }  **public** **void** setBrand\_cname(String brand\_cname) {  **this**.brand\_cname = brand\_cname;  }    } |

## Yml

|  |
| --- |
| server:  port: 8888  eureka:  client:  serviceUrl:  defaultZone: http://localhost:666/eureka/  spring:  application:  name: go9-mall  thymeleaf:  cache: **false**  logging:  level:  root: info  #custom properties  #LOGIN\_URL: http://localhost:3030/login.html  #index roll api  index:  roll:  url: http://GO9-GOODS-SERVICE/admin/indexroll    #search  search:  url: http://go9-search-service/search |

## List.html

|  |
| --- |
| <ul class=*"list"*>  <li th:each=*"product : ${sr.records}"*>  <div class=*"gPic"*>  <img th:src=*"@{${product.picture}}"* alt=*""*>  </div>  <h5 th:utext=*"${product.title}"* class=*"title"*></h5>  <div class=*"switch"*>  <p th:text=*"'￥'+${product.price/100}"* class=*"price"* style="color:*red*;"></p>  <p class=*"btns"*>  <a class=*"goDetail"* th:href=*"@{/product.html(id=${product.id})}"*>查看详情</a>  </p>  </div>  </li>  </ul> |

|  |
| --- |
| <!-- 分页 -->  <div id=*"pagination"* style="text-align: *center*; margin: *30px 0*">  <Page :total=*"total"* :page-size=*"60"* @on-change=*"onPageChange"* />  </div> |

|  |
| --- |
| **var** vuePage = **new** Vue({  el: "#pagination",  data: {  total: /\*[[${sr.total}]]\*/ 0  },  computed: {    },  methods: {  onPageChange(page){  window.location.href = "http://localhost:8888/list.html?q="+**this**.searchData+"&page="+page  }  }  }) |