# 商品搜索

ES api

|  |
| --- |
| GET /usian/item/\_search  {  "query": {  "multi\_match": {  "query": "a",  "fields": ["item\_title","item\_sell\_point","item\_category\_name","item\_desc"]  }  },  "from": 0,  "size": 20,  "highlight": {  "pre\_tags": ["<font style='color:red'>"],  "post\_tags": ["</font>"],  "fields": {  "item\_title": {}  }  }  }  } |

usian\_search\_service

Com.usian.service.SearchItemServiceImpl

|  |
| --- |
| /\*\*  \* 商品搜索  \* @param q  \* @param page  \* @param rows  \* @return  \*/  @Override  public List<SearchItem> list(String q, Long page, Integer rows) {  try {  SearchRequest searchRequest = new SearchRequest(ES\_INDEX\_NAME);  searchRequest.types(ES\_TYPE\_NAME);  SearchSourceBuilder searchSourceBuilder = new SearchSourceBuilder();  searchSourceBuilder.query(QueryBuilders.multiMatchQuery(q,  new String[]{"item\_title", "item\_sell\_point", "item\_category\_name", "item\_desc"}));  Long from = (page - 1) \* rows;  searchSourceBuilder.from(from.intValue());  searchSourceBuilder.size(rows);  //高亮  HighlightBuilder highlightBuilder = new HighlightBuilder();  highlightBuilder.preTags("<font color='red'>");  highlightBuilder.postTags("</font>");  highlightBuilder.field("item\_title");  searchSourceBuilder.highlighter(highlightBuilder);  searchRequest.source(searchSourceBuilder);  SearchResponse searchResponse =  restHighLevelClient.search(searchRequest, RequestOptions.DEFAULT);  SearchHit[] hits = searchResponse.getHits().getHits();  List<SearchItem> searchItems = new ArrayList<>();  for (int i = 0; i < hits.length; i++) {  SearchHit hit = hits[i];  SearchItem searchItem = JsonUtils.jsonToPojo(hit.getSourceAsString(), SearchItem.class);  Map<String, HighlightField> highlightFields = hit.getHighlightFields();  if (highlightFields != null && highlightFields.size() > 0) {  searchItem.setItem\_title(highlightFields.get("item\_title").  getFragments()[0].toString());  }  searchItems.add(searchItem);  }  return searchItems;  } catch (IOException e) {  e.printStackTrace();  }  return null;  } |

Usian\_search\_feign

|  |
| --- |
| @RequestMapping("/service/searchItem/list")  List<SearchItem> selectByq(@RequestParam String q, @RequestParam Long page,  @RequestParam Integer pageSize); |

Usian\_search\_web

Com.usian.controller.SearchItemController

|  |
| --- |
| @RequestMapping("/list")  public List<SearchItem> selectByQ(String q, @RequestParam(defaultValue = "1")  Long page, @RequestParam(defaultValue = "20") Integer pageSize){  return searchItemFeign.selectByq(q,page,pageSize);  } |

# 索引库同步

## 分析

**方案一**：业务逻辑在usian\_item\_sevice中实现，添加商品的业务逻辑中，添加一个同步索引库的业务逻辑。

缺点：

​ 1、业务逻辑耦合度高（既维护商品又维护索引库）

**方案二**：业务逻辑在usian\_search\_service中实现，usian\_item\_sevice先添加商品，再调用usian\_search\_service服务同步索引库。

缺点：

​ 1、系统间耦合性太强（如果将来其他服务接入，usian\_item\_sevice还需要修改代码）

**方案三**：使用消息中间件

Rabbitmq 解耦，异步，削峰

| **类型** | **特点** |
| --- | --- |
| 基本消息模型 | 发送者------>queue----->消费者 |
| work消息模型 | 发送者------>queue----->多个消费者 |
| 广播-fanout消息模型 | 发送者----->exchange---->多个queue--->多个消费者 |
| 定向-direct消息模式 | 发送者----->exchange---routing key-->多个queue--->多个消费者 |
| 通配符-topic消息模型 | 发送者----->exchange---星.routing key.星-->多个queue---->多个消费者 |

## Usian\_item\_service

Pom.xml

|  |
| --- |
| <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-amqp</artifactId>  </dependency> |

Application.yml

|  |
| --- |
| spring:  rabbitmq:  host: 192.168.233.132  port: 5672  username: admin  password: 1111  virtual-host: / |

Com.usian.service.ItemserviceImpl

|  |
| --- |
| //添加商品发布消息到mq  amqpTemplate.convertAndSend("item\_exchage","item.add", itemId); |

## Usian\_search\_service

Pom.xml

|  |
| --- |
| <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-amqp</artifactId>  </dependency> |

Application.yml

|  |
| --- |
| spring:  rabbitmq:  host: 192.168.233.132  port: 5672  username: admin  password: 1111  virtual-host: / |

Com.usin.service.SearchItemService

|  |
| --- |
| @Override  public int insertDocument(String itemId) throws IOException {  // 1、根据商品id查询商品信息。  SearchItem searchItem = searchItemMapper.getItemById(Long.valueOf(itemId));  //2、添加商品到索引库  IndexRequest indexRequest = new IndexRequest(ES\_INDEX\_NAME, ES\_TYPE\_NAME);  indexRequest.source(JsonUtils.objectToJson(searchItem), XContentType.JSON);  IndexResponse indexResponse =  restHighLevelClient.index(indexRequest,RequestOptions.DEFAULT);  return indexResponse.getShardInfo().getFailed();  } |

Com.usian.listener.SearchListener

|  |
| --- |
| /\*\*  \* 监听者接收消息三要素：  \* 1、queue  \* 2、exchange  \* 3、routing key  \*/  @RabbitListener(bindings = @QueueBinding(  value = @Queue(value="search\_queue",durable = "true"),  exchange = @Exchange(value="item\_exchage",type= ExchangeTypes.TOPIC),  key= {"item.\*"}  ))  public void listen(String msg) throws Exception {  System.out.println("接收到消息：" + msg);  int result = searchItemService.insertDocument(msg);  if(result>0){  throw new RuntimeException("同步失败");  }  }  } |