

# 学成在线-第11天-讲义-搜索服务

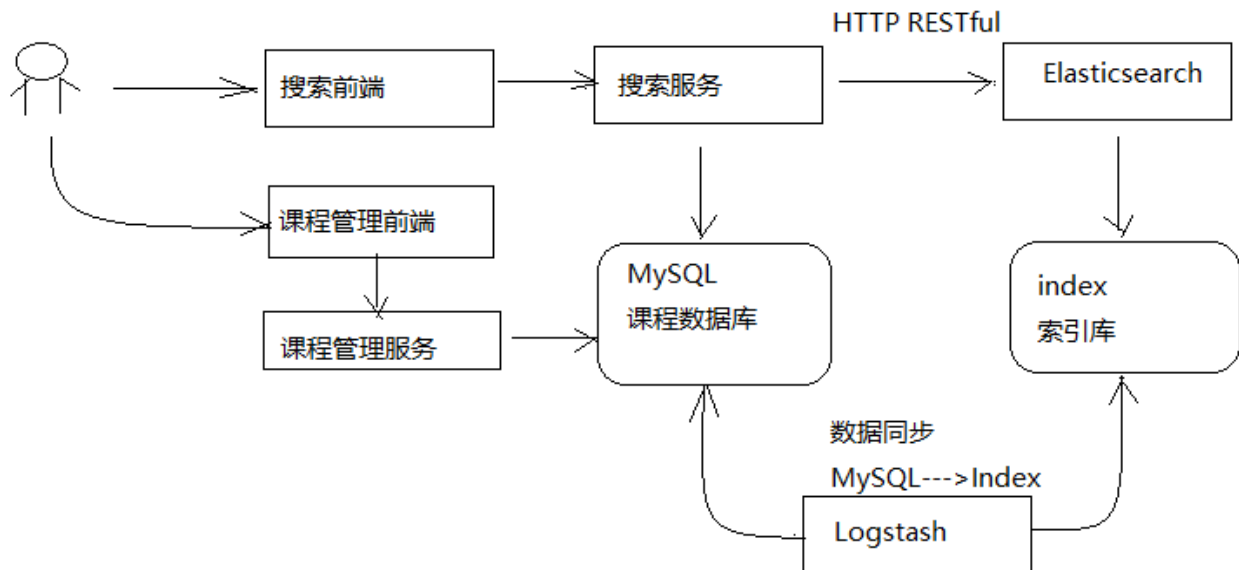
## 1 课程搜索需求分析

### 1.1 需求分析



- 1、根据分类搜索课程信息。
- 2、根据关键字搜索课程信息，搜索方式为全文检索，关键字需要匹配课程的名称、课程内容。这个不是很必要吧？
- 3、根据难度等级搜索课程。
- 4、搜索结点分页显示。

### 1.2 搜索流程



- 1、课程管理服务将数据写到MySQL数据库
- 2、使用Logstash将MySQL数据库中的数据写到ES的索引库。
- 3、用户在前端搜索课程信息，请求到搜索服务。
- 4、搜索服务请求ES搜索课程信息。

## 2 全文检索技术研究

参考：elasticsearch研究.md

研究ElasticSearch搜索方法。

## 3 课程索引

### 3.1 技术方案

如何维护课程索引信息？

- 1、当课程向MySQL添加后同时将课程信息添加到索引库。  
采用Logstash实现，Logstash会从MySQL中将数据采集到ES索引库。
- 2、当课程在MySQL更新信息后同时更新该课程在索引库的信息。  
采用Logstash实现。
- 3、当课程在MySQL删除后同时将该课程从索引库删除。  
手工写程序实现，在删除课程后将索引库中该课程信息删除。

## 3.2 准备课程索引信息

课程发布成功在MySQL数据库存储课程发布信息，此信息作为课程索引信息。

### 3.2.1 创建课程发布表

课程信息分布在course\_base、course\_pic等不同的表中。

课程发布成功为了方便进行索引将这几张表的数据合并在一张表中，作为课程发布信息。

创建course\_pub表

	Field	Type	Comment
	id	varchar(32) NOT NULL	主键
	name	varchar(32) NOT NULL	课程名称
	users	varchar(500) NOT NULL	适用人群
	mt	varchar(32) NOT NULL	大分类
	st	varchar(32) NOT NULL	小分类
	grade	varchar(32) NOT NULL	课程等级
	studymodel	varchar(32) NOT NULL	学习模式
	teachmode	varchar(32) NULL	教育模式
	description	text NOT NULL	课程介绍
	timestamp	timestamp NOT NULL	时间戳logstash使用
	charge	varchar(32) NOT NULL	收费规则，对应数据字典
	valid	varchar(32) NOT NULL	有效性，对应数据字典
	qq	varchar(32) NULL	咨询qq
	price	float(10,2) NULL	价格
	price_old	float(10,2) NULL	原价格
	expires	varchar(32) NULL	过期时间
	start_time	varchar(32) NULL	课程有效期-开始时间
	end_time	varchar(32) NULL	课程有效期-结束时间
	pic	varchar(500) NULL	课程图片
	teachplan	text NOT NULL	课程计划
	pub_time	varchar(32) NULL	发布时间

### 3.2.2 创建课程发布表模型

在课程管理服务创建模型：

```
@Data
@ToString
@Entity
@Table(name="course_pub")
@GenericGenerator(name = "jpa-assigned", strategy = "assigned")
public class CoursePub implements Serializable {
    private static final long serialVersionUID = -916357110051689487L;
    @Id
    @GeneratedValue(generator = "jpa-assigned")
    @Column(length = 32)

    private String id;
```



```
private String name;
private String users;
private String mt;
private String st;
private String grade;
private String studymodel;
private String teachmode;
private String description;
private String pic;//图片
private Date timestamp;//时间戳
private String charge;
private String valid;
private String qq;
private Float price;
private Float price_old;
private String expires;
private String teachplan;//课程计划
@Column(name="pub_time")
private String pubTime;//课程发布时间
}
```

### 3.2.3修改课程发布

在课程管理服务定义dao：

1) 创建course\_pub表的dao

```
public interface CoursePubRepository extends JpaRepository<CoursePub, String> {
}
```

2) 修改课程发布service

```
//保存CoursePub
public CoursePub saveCoursePub(String id, CoursePub coursePub){
    if(StringUtils.isEmpty(id)){
        ExceptionCast.cast(CourseCode.COURSE_PUBLISH_COURSEIDISNULL);
    }
    CoursePub coursePubNew = null;
    Optional<CoursePub> coursePubOptional = coursePubRepository.findById(id);
    if(coursePubOptional.isPresent()){
        coursePubNew = coursePubOptional.get();
    }
    if(coursePubNew == null){
        coursePubNew = new CoursePub();
    }

    BeanUtils.copyProperties(coursePub,coursePubNew);
    //设置主键
    coursePubNew.setId(id); 重新设置以防没有
    //更新时间戳为最新时间
    coursePubNew.setTimestamp(new Date()); logstash要用
    //发布时间
```



```
SimpleDateFormat simpleDateFormat = new SimpleDateFormat("YYYY-MM-dd HH:mm:ss");
String date = simpleDateFormat.format(new Date());
coursePub.setPubTime(date);          字符串
coursePubRepository.save(coursePub);
return coursePub;

}
//创建coursePub对象
private CoursePub createCoursePub(String id){
    CoursePub coursePub = new CoursePub();
    coursePub.setId(id);

    //基础信息
    Optional<CourseBase> courseBaseOptional = courseBaseRepository.findById(id);
    if(courseBaseOptional == null){
        CourseBase courseBase = courseBaseOptional.get();
        BeanUtils.copyProperties(courseBase, coursePub);
    }
    //查询课程图片
    Optional<CoursePic> picOptional = coursePicRepository.findById(id);
    if(picOptional.isPresent()){
        CoursePic coursePic = picOptional.get();
        BeanUtils.copyProperties(coursePic, coursePub);
    }

    //课程营销信息
    Optional<CourseMarket> marketOptional = courseMarketRepository.findById(id);
    if(marketOptional.isPresent()){
        CourseMarket courseMarket = marketOptional.get();
        BeanUtils.copyProperties(courseMarket, coursePub);
    }

    //课程计划
    TeachplanNode teachplanNode = teachplanMapper.selectList(id);
    //将课程计划转成json
    String teachplanString = JSON.toJSONString(teachplanNode);
    coursePub.setTeachplan(teachplanString);
    return coursePub;
}
```

修改课程发布方法，添加调用saveCoursePub方法的代码，添加部分的代码如下：

```
//课程发布
@Transactional
public CoursePublishResult publish(String courseId){
    ....
    //创建课程索引
    //创建课程索引信息
    CoursePub coursePub = createCoursePub(courseId);
    //向数据库保存课程索引信息

    CoursePub newCoursePub = saveCoursePub(courseId, coursePub);
}
```

```
        if(newCoursePub==null){
            //创建课程索引信息失败
            ExceptionCast.cast(CourseCode.COURSE_PUBLISH_CREATE_INDEX_ERROR);
        }
        ....
    }
```

## 3.3 搭建ES环境

### 3.3.1 ES安装

开发环境使用ES单机环境，启动ES服务端。

注意：旧的ES环境，可以删除elasticsearch-6.2.1\data\nodes目录以完全清除ES环境。

安装elasticsearch-head并启动。

### 3.3.2 创建索引库

创建索引库

创建xc\_course索引库，一个分片，0个副本。

### 3.3.3 创建映射

Post [http://localhost:9200/xc\\_course/doc/ mapping](http://localhost:9200/xc_course/doc/ mapping)

```
{
  "properties" : {

    "description" : {
      "analyzer" : "ik_max_word",
      "search_analyzer": "ik_smart",
      "type" : "text"
    },
    "grade" : {
      "type" : "keyword"
    },
    "id" : {
      "type" : "keyword"
    },
    "mt" : {
      "type" : "keyword"
    },
    "name" : {
```

```
        "analyzer" : "ik_max_word",
        "search_analyzer": "ik_smart",
        "type" : "text"
    },
    "users" : {
        "index" : false,
        "type" : "text"
    },
    "charge" : {
        "type" : "keyword"
    },
    "valid" : {
        "type" : "keyword"
    },
    "pic" : {
        "index" : false,
        "type" : "keyword"
    },
    "qq" : {
        "index" : false,
        "type" : "keyword"
    },
    "price" : {
        "type" : "float"
    },
    "price_old" : {
        "type" : "float"
    },
    "st" : {
        "type" : "keyword"
    },
    "status" : {
        "type" : "keyword"
    },
    "studymodel" : {
        "type" : "keyword"
    },
    "teachmode" : {
        "type" : "keyword"
    },
    "teachplan" : {
        "analyzer" : "ik_max_word",
        "search_analyzer": "ik_smart",
        "type" : "text"
    },
    "expires" : {
        "type" : "date",
        "format": "yyyy-MM-dd HH:mm:ss"
    },
    "pub_time" : {
        "type" : "date",
        "format": "yyyy-MM-dd HH:mm:ss"
```

```
    },  
    "start_time" : {  
      "type" : "date",  
      "format": "yyyy-MM-dd HH:mm:ss"  
    },  
    "end_time" : {  
      "type" : "date",  
      "format": "yyyy-MM-dd HH:mm:ss"  
    }  
  }  
}
```

## 3.4 Logstash创建索引

Logstash是ES下的一款开源软件，它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Elasticsearch中创建索引。

本项目使用Logstash将MySQL中的数据采用到ES索引中。

### 3.4.1 下载Logstash

下载Logstash6.2.1版本，和本项目使用的Elasticsearch6.2.1版本一致。

<https://www.elastic.co/downloads/logstash>

The screenshot shows the Elastic website's download page for Logstash. At the top, the Elastic logo is on the left, and navigation links for Products, Cloud, Services, Customers, and Learn are in the center. On the right, there are links for downloads and a contact button. Below the navigation bar, the page title is "Download Logstash". A prominent blue button with a circular arrow icon says "Want to upgrade? We'll give you a hand. Migration Guide »". Below this, the version "6.2.1" is displayed. The release date is "February 08, 2018". The notes section includes links to "release notes" and "past releases", and a statement that "Java 8 is required for Logstash 6.x and 5.x". At the bottom, under "Downloads:", there are links for TAR.GZ, ZIP, DEB, and RPM, each followed by a "sha" link.

解压：



ata (D:) > Elastic > logstash-6.2.1 >

名称
bin
config
data
lib
logstash-core
logstash-core-plugin-api
modules
tools
vendor
CONTRIBUTORS
Gemfile
Gemfile.lock
LICENSE
NOTICE.TXT

### 3.4.2 安装logstash-input-jdbc

logstash-input-jdbc 是ruby开发的，先下载ruby并安装

下载地址: <https://rubyinstaller.org/downloads/>

下载2.5版本即可。

安装完成查看是否安装成功

```
C:\Users\admin>ruby -v  
ruby 2.5.0p0 (2017-12-25 revision 61468) [x64-mingw32]
```

Logstash5.x以上版本本身自带logstash-input-jdbc，6.x版本本身不带logstash-input-jdbc插件，需要手动安装

```
PS D:\Elastic\logstash-6.2.1\bin> .\logstash-plugin.bat install logstash-input-jdbc  
Validating logstash-input-jdbc  
Installing logstash-input-jdbc  
Installation successful
```

安装成功后我们可以在logstash根目录下的以下目录查看对应的插件版本

> Data (D:) > Elastic > logstash-6.2.1 > vendor > bundle > jruby > 2.3.0 > gems >

名称	修改日期	类型
logstash-input-http-3.0.8	2018/2/18 20:44	文件夹
logstash-input-imap-3.0.5	2018/2/18 20:44	文件夹
logstash-input-jdbc-4.3.3	2018/2/18 20:44	文件夹

解压老师提供的logstash-6.2.1.zip,此logstash中已集成了logstash-input-jdbc插件。

### 3.4.3 创建模板文件

Logstash的工作是从MySQL中读取数据，向ES中创建索引，这里需要提前创建mapping的模板文件以便logstash使用。

在logstash的config目录创建xc\_course\_template.json，内容如下：

本教程的xc\_course\_template.json目录是：D:/ElasticSearch/logstash-6.2.1/config/xc\_course\_template.json

```
{
  "mappings" : {
    "doc" : {
      "properties" : {
        "charge" : {
          "type" : "keyword"
        },
        "description" : {
          "analyzer" : "ik_max_word",
          "search_analyzer" : "ik_smart",
          "type" : "text"
        },
        "end_time" : {
          "format" : "yyyy-MM-dd HH:mm:ss",
          "type" : "date"
        },
        "expires" : {
          "format" : "yyyy-MM-dd HH:mm:ss",
          "type" : "date"
        },
        "grade" : {
          "type" : "keyword"
        },
        "id" : {
          "type" : "keyword"
        },
        "mt" : {
          "type" : "keyword"
        },
        "name" : {
```

```
        "analyzer" : "ik_max_word",
        "search_analyzer" : "ik_smart",
        "type" : "text"
    },
    "pic" : {
        "index" : false,
        "type" : "keyword"
    },
    "price" : {
        "type" : "float"
    },
    "price_old" : {
        "type" : "float"
    },
    "pub_time" : {
        "format" : "yyyy-MM-dd HH:mm:ss",
        "type" : "date"
    },
    "qq" : {
        "index" : false,
        "type" : "keyword"
    },
    "st" : {
        "type" : "keyword"
    },
    "start_time" : {
        "format" : "yyyy-MM-dd HH:mm:ss",
        "type" : "date"
    },
    "status" : {
        "type" : "keyword"
    },
    "studymodel" : {
        "type" : "keyword"
    },
    "teachmode" : {
        "type" : "keyword"
    },
    "teachplan" : {
        "analyzer" : "ik_max_word",
        "search_analyzer" : "ik_smart",
        "type" : "text"
    },
    "users" : {
        "index" : false,
        "type" : "text"
    },
    "valid" : {
        "type" : "keyword"
    }
}

},
```

```
"template" : "xc_course"
}
```

### 3.4.4 配置mysql.conf

在logstash的config目录下配置mysql.conf文件供logstash使用，logstash会根据mysql.conf文件的配置的地址从MySQL中读取数据向ES中写入索引。

参考<https://www.elastic.co/guide/en/logstash/current/plugins-inputs-jdbc.html>

配置输入数据源和输出数据源。

```
input {
  stdin {
    a}
  jdbc {
    jdbc_connection_string => "jdbc:mysql://localhost:3306/xc_course?
useUnicode=true&characterEncoding=utf-8&useSSL=true&serverTimezone=UTC"
    # the user we wish to excute our statement as
    jdbc_user => "root"
    jdbc_password => mysql 数字需要用双引号括起来, 不然会报异常
    # the path to our downloaded jdbc driver
    jdbc_driver_library => "F:/develop/maven/repository3/mysql/mysql-connector-java/5.1.41/mysql-
connector-java-5.1.41.jar"
    # the name of the driver class for mysql
    jdbc_driver_class => "com.mysql.jdbc.Driver"
    jdbc_paging_enabled => "true"
    jdbc_page_size => "50000"
    #要执行的sql文件
    #statement_filepath => "/conf/course.sql"
    statement => "select * from course_pub where timestamp > date_add(:sql_last_value,INTERVAL 8
HOUR)"
    #定时配置
    schedule => "* * * * *"
    record_last_run => true
    last_run_metadata_path => "D:/ElasticSearch/logstash-6.2.1/config/logstash_metadata"
  }
}

output {
  elasticsearch {
    #ES的ip地址和端口
    hosts => "localhost:9200"
    #hosts => ["localhost:9200","localhost:9202","localhost:9203"]
    #ES索引库名称
    index => "xc_course"
    document_id => "%{id}"
  }
}
```

Unable to connect to database. Tried 1 times  
{:error\_message=>"TypeError: could not coerce  
Fixnum to class java.lang.String"}

```
document_type => "doc"
template => "D:/ElasticSearch/logstash-6.2.1/config/xc_course_template.json"
template_name => "xc_course"
template_overwrite => "true"
}
stdout {
#日志输出
codec => json_lines
}
}
```

说明：

### 1、ES采用UTC时区问题

ES采用UTC 时区，比北京时间早8小时，所以ES读取数据时让最后更新时间加8小时

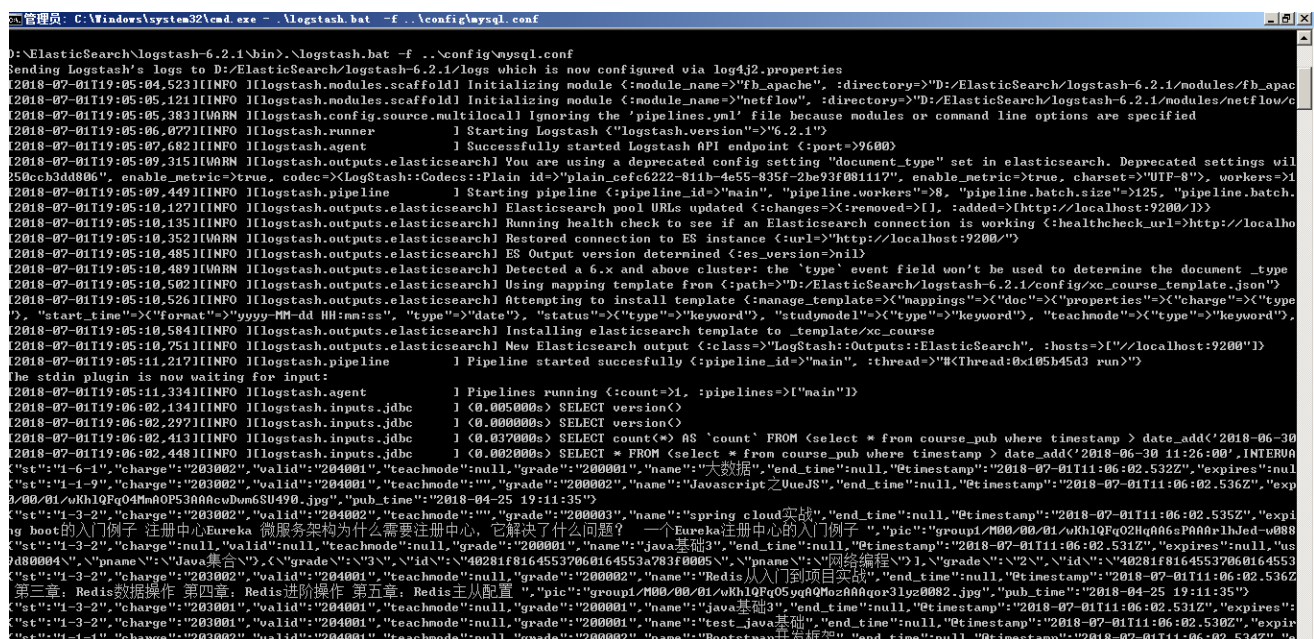
where timestamp > date\_add(sql\_last\_value,INTERVAL 8 HOUR)

2、logstash每个执行完成会在D:/ElasticSearch/logstash-6.2.1/config/logstash\_metadata记录执行时间下次以此时间为基准进行增量同步数据到索引库。

## 3.4.5 测试

启动logstash.bat：

```
cd E:\ElasticSearch\logstash-6.2.1\bin
.\logstash.bat -f ..\config\mysql.conf
```



修改course\_pub中的数据，并且修改timestamp为当前时间，观察Logstash日志是否读取到要索引的数据。

最后用head登录ES查看索引文档内容是否修改。

Elasticsearch http://localhost:9200/ 连接 xuecheng-search 当前时间: green (1 of 1)

索引: xuecheng\_index

类型: xc\_course

字段: @timestamp, @version, charge

查询 1 个分片中用的 1 个, 7 命中, 耗时 0.000 秒

id	name	users	valid	teachplan
4028e58161bd3b380161bd3bcd2f0000	Redis从入门到项目实战		204001	第一章: redis简介 第一节 NoSQL简介 第二节 认识Redis 第二章:
4028e58161bcb7f40161bcb77c0000	spring cloud实战	所有人	204002	微服务架构入门 为什么要使用微服务: 单体架构的特点 为什么要使用
4028e58161bd22e60161bd23672a0001	Javascript之VueJS	所有人	204001	Vuejs 第一讲 第一节 vue基础, 常用指令, bootstrap+vue的简介
4028858162e0bc0a0162e0bdf1a0000	人工智能+python	小白	204002	
402885816243d2dd016243f24c030002	大数据	具有一定的java基础	204001	
4028e581617f945f01617f9dabc40000	Bootstrap开发框架		204001	计算机原理 计算机硬件 计算机软件 计算机编程入门 java语法介绍
297e7c7c62b88f00162b8a965510001	test_java基础	test_java基础	204001	java基础语法

## 4 课程搜索

### 4.1 需求分析

学成在线 在线教育·学有所成

首页 课程 职业规划

输入查询关键词 搜索

我的学习 退出 教学提醒

关键字:

一级分类: 全部 前端开发 移动开发 编程开发 数据库 人工智能 云计算/大数据 UI设计 游戏开发 智能硬件/物联网 研发管理

二级分类: 全部

难度等级: 全部 初级 中级 高级

猜你喜欢

spring cloud实战 高级 · 1125人在学习

Javascript之VueJS 高级 · 1125人在学习

Bootstrap开发框架 高级 · 1125人在学习

Redis从入门到项目实战 高级 · 1125人在学习

通过对ThinkPHP框架基础, 带领大家由浅入深轻松掌握ThinkPHP的理论基础, 更加全面的掌握ThinkPHP框架运行机制.....

Think PHP 5.0 博客系统实战项目演练

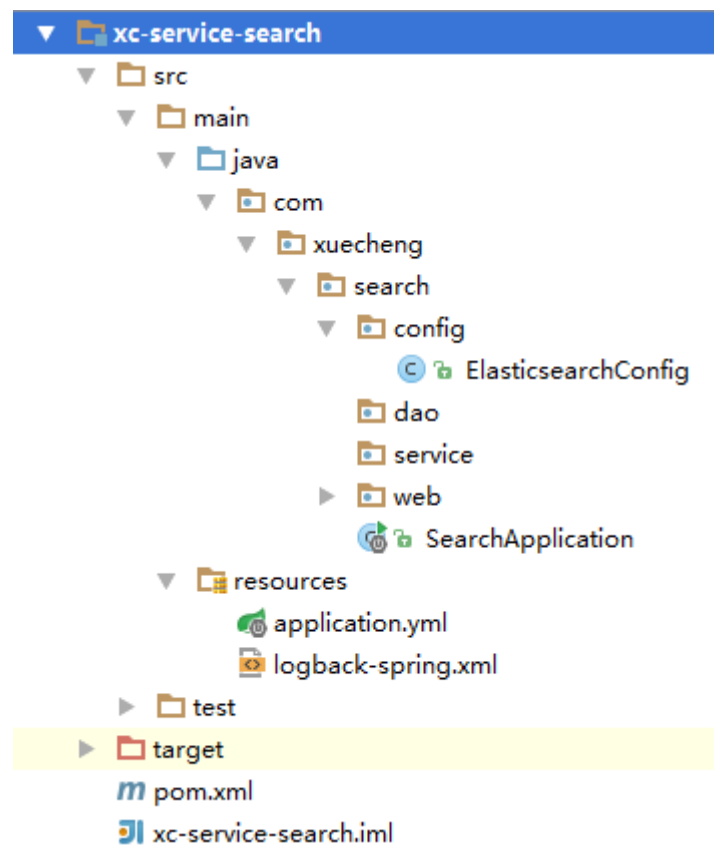
- 1、根据分类搜索课程信息。
- 2、根据关键字搜索课程信息, 搜索方式为全文检索, 关键字需要匹配课程的名称、 课程内容。
- 3、根据难度等级搜索课程。
- 4、搜索结点分页显示。

技术分析:

- 1、根据关键字搜索, 采用MultiMatchQuery, 搜索name、description、teachplan
- 2、根据分类、课程等级搜索采用过滤器实现。
- 3、分页查询。
- 4、高亮显示。

### 4.2 创建搜索服务工程

## 1) 创建xc-service-search工程



## 2) 配置

### 1、配置application.yml

```
server:
  port: 40100
spring:
  application:
    name: xc-search-service
elasticsearch:
  hostlist: 127.0.0.1:9200 #多个结点中间用逗号分隔
course:
  index: xc_course
  type: doc
```

### 2、配置RestHighLevelClient和RestClient

```
package com.xuecheng.search.config;

import org.apache.http.HttpHost;
import org.elasticsearch.client.RestClient;
import org.elasticsearch.client.RestHighLevelClient;
import org.springframework.beans.factory.annotation.Value;
```

```
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class ElasticsearchConfig {

    @Value("${xuecheng.elasticsearch.hostlist}")
    private String hostlist;

    @Bean
    public RestHighLevelClient restHighLevelClient(){
        //解析hostlist配置信息
        String[] split = hostlist.split(",");
        //创建HttpHost数组，其中存放es主机和端口的配置信息
        HttpHost[] httpHostArray = new HttpHost[split.length];
        for(int i=0;i<split.length;i++){
            String item = split[i];
            httpHostArray[i] = new HttpHost(item.split(":")[0], Integer.parseInt(item.split(":")
[1]), "http");
        }
        //创建RestHighLevelClient客户端
        return new RestHighLevelClient(RestClient.builder(httpHostArray));
    }

    @Bean
    public RestClient restClient(){
        //解析hostlist配置信息
        String[] split = hostlist.split(",");
        //创建HttpHost数组，其中存放es主机和端口的配置信息
        HttpHost[] httpHostArray = new HttpHost[split.length];
        for(int i=0;i<split.length;i++){
            String item = split[i];
            httpHostArray[i] = new HttpHost(item.split(":")[0], Integer.parseInt(item.split(":")
[1]), "http");
        }
        return RestClient.builder(httpHostArray).build();
    }
}
```

## 4.3 API



```
@Api(value = "课程搜索",description = "课程搜索",tags = {"课程搜索"})
public interface EsCourseControllerApi {

    @ApiOperation("课程搜索")
    public QueryResponseResult<CoursePub> list(int page,int size,
        CourseSearchParams courseSearchParams) throws IOException;

}
```

## 4.4 Service

Service方法代码复杂，这里分三步完成。

### 4.4.1 按关键字搜索

1) 在application.yml中配置source\_field

```
elasticsearch:
  hostlist: 127.0.0.1:9200 #多个结点中间用逗号分隔
  course:
    index: xc_course
    type: doc
    source_field:
      id,name,grade,mt,st,charge,valid,pic,qq,price,price_old,status,studymodel,teachmode,expires,pub_
      time,start_time,end_time
```

2) service完整代码如下

```
@Service
public class EsCourseService {
    private static final Logger LOGGER = LoggerFactory.getLogger(EsCourseService.class);

    @Value("${xuecheng.elasticsearch.course.index}")
    private String es_index;
    @Value("${xuecheng.elasticsearch.course.type}")
    private String es_type;
    @Value("${xuecheng.elasticsearch.course.source_field}")
    private String source_field;

    @Autowired
    RestHighLevelClient restHighLevelClient;

    public QueryResponseResult<CoursePub> list(int page,int size,CourseSearchParams
```



```
courseSearchParam) {

    //设置索引
    SearchRequest searchRequest = new SearchRequest(es_index);
    //设置类型
    searchRequest.types(es_type);
    SearchSourceBuilder searchSourceBuilder = new SearchSourceBuilder();
    BoolQueryBuilder boolQueryBuilder = QueryBuilders.boolQuery();
    //source源字段过滤
    String[] source_fields = source_field.split(",");
    searchSourceBuilder.fetchSource(source_fields, new String[]{});
    //关键字
    if(StringUtils.isEmpty(courseSearchParam.getKeyword())){
        //匹配关键字
        MultiMatchQueryBuilder multiMatchQueryBuilder =
            QueryBuilders.multiMatchQuery(courseSearchParam.getKeyword(), "name",
            "teachplan","description");
        //设置匹配占比
        multiMatchQueryBuilder.minimumShouldMatch("70%");
        //提升另一个字段的Boost值
        multiMatchQueryBuilder.field("name",10);
        boolQueryBuilder.must(multiMatchQueryBuilder);
    }

    //布尔查询
    searchSourceBuilder.query(boolQueryBuilder);

    //请求搜索
    searchRequest.source(searchSourceBuilder);
    SearchResponse searchResponse = null;
    try {
        searchResponse = restHighLevelClient.search(searchRequest);
    } catch (IOException e) {
        e.printStackTrace();
        LOGGER.error("xuecheng search error..{}",e.getMessage());
        return new QueryResponseResult(CommonCode.SUCCESS,new QueryResult<CoursePub>());
    }

    //结果集处理
    SearchHits hits = searchResponse.getHits();
    SearchHit[] searchHits = hits.getHits();
    //记录总数
    long totalHits = hits.getTotalHits();
    //数据列表
    List<CoursePub> list = new ArrayList<>();

    for (SearchHit hit : searchHits) {
        CoursePub coursePub = new CoursePub();

        //取出source
        Map<String, Object> sourceAsMap = hit.getSourceAsMap();

        //取出名称
```

```
String name = (String) sourceAsMap.get("name");
coursePub.setName(name);
//图片
String pic = (String) sourceAsMap.get("pic");
coursePub.setPic(pic);
//价格
Float price = null;
try {
    if(sourceAsMap.get("price")!=null ){
        price = Float.parseFloat((String) sourceAsMap.get("price"));
    }

} catch (Exception e) {
    e.printStackTrace();
}
coursePub.setPrice(price);
Float price_old = null;
try {
    if(sourceAsMap.get("price_old")!=null ){
        price_old = Float.parseFloat((String) sourceAsMap.get("price_old"));
    }
} catch (Exception e) {
    e.printStackTrace();
}
coursePub.setPrice_old(price_old);

list.add(coursePub);

}

QueryResult<CoursePub> queryResult = new QueryResult<>();
queryResult.setList(list);
queryResult.setTotal(totalHits);
QueryResponseResult<CoursePub> coursePubQueryResponseResult = new
QueryResponseResult<CoursePub>(CommonCode.SUCCESS,queryResult);
return coursePubQueryResponseResult;
}

}
```

## 4.4.2 按分类和难度等级搜索

按分类和难度等级搜索代码如下：

```
@Service
public class EsCourseService {
    private static final Logger LOGGER = LoggerFactory.getLogger(EsCourseService.class);

    @Value("${xuecheng.elasticsearch.course.index}")
    private String es_index;
    @Value("${xuecheng.elasticsearch.course.type}")
    private String es_type;
```



```
@Value("${xuecheng.elasticsearch.course.source_field}")
private String source_field;

@Autowired
RestHighLevelClient restHighLevelClient;

public QueryResponseResult<CoursePub> list(int page,int size,CourseSearchParam
courseSearchParam) {

    //设置索引
    SearchRequest searchRequest = new SearchRequest(es_index);
    //设置类型
    searchRequest.types(es_type);
    SearchSourceBuilder searchSourceBuilder = new SearchSourceBuilder();
    BoolQueryBuilder boolQueryBuilder = QueryBuilders.boolQuery();
    //source源字段过滤
    String[] source_fields = source_field.split(",");
    searchSourceBuilder.fetchSource(source_fields, new String[]{});
    //关键字
    if(StringUtils.isNotEmpty(courseSearchParam.getKeyword())){
        //匹配关键字
        MultiMatchQueryBuilder multiMatchQueryBuilder =
        QueryBuilders.multiMatchQuery(courseSearchParam.getKeyword(), "name",
        "teachplan","description");
        //设置匹配占比
        multiMatchQueryBuilder.minimumShouldMatch("70%");
        //提升另一个字段的Boost值
        multiMatchQueryBuilder.field("name",10);
        boolQueryBuilder.must(multiMatchQueryBuilder);
    }
    //过滤
    if(StringUtils.isNotEmpty(courseSearchParam.getMt())){
        boolQueryBuilder.filter(QueryBuilders.termQuery("mt",courseSearchParam.getMt()));
    }
    if(StringUtils.isNotEmpty(courseSearchParam.getSt())){
        boolQueryBuilder.filter(QueryBuilders.termQuery("st",courseSearchParam.getSt()));
    }
    if(StringUtils.isNotEmpty(courseSearchParam.getGrade())){
        boolQueryBuilder.filter(QueryBuilders.termQuery("grade",courseSearchParam.getGrade()));
    }
    //布尔查询
    searchSourceBuilder.query(boolQueryBuilder);

    //请求搜索
    searchRequest.source(searchSourceBuilder);
    SearchResponse searchResponse = null;
    try {
        searchResponse = restHighLevelClient.search(searchRequest);
    } catch (IOException e) {
        e.printStackTrace();
        LOGGER.error("xuecheng search error..{}",e.getMessage());
    }

    return new QueryResponseResult(CommonCode.SUCCESS,new QueryResult<CoursePub>());
}
```



```
}

//结果集处理
SearchHits hits = searchResponse.getHits();
SearchHit[] searchHits = hits.getHits();
//记录总数
long totalHits = hits.getTotalHits();
//数据列表
List<CoursePub> list = new ArrayList<>();

for (SearchHit hit : searchHits) {
    CoursePub coursePub = new CoursePub();

    //取出source
    Map<String, Object> sourceAsMap = hit.getSourceAsMap();

    //取出名称
    String name = (String) sourceAsMap.get("name");

    coursePub.setName(name);
    //图片
    String pic = (String) sourceAsMap.get("pic");
    coursePub.setPic(pic);
    //价格
    Float price = null;
    try {
        if(sourceAsMap.get("price")!=null ){
            price = Float.parseFloat((String) sourceAsMap.get("price"));
        }

    } catch (Exception e) {
        e.printStackTrace();
    }
    coursePub.setPrice(price);
    Float price_old = null;
    try {
        if(sourceAsMap.get("price_old")!=null ){
            price_old = Float.parseFloat((String) sourceAsMap.get("price_old"));
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    coursePub.setPrice_old(price_old);

    list.add(coursePub);
}

QueryResult<CoursePub> queryResult = new QueryResult<>();
queryResult.setList(list);
queryResult.setTotal(totalHits);
QueryResponseResult<CoursePub> coursePubQueryResponseResult = new
QueryResponseResult<CoursePub>(CommonCode.SUCCESS,queryResult);

return coursePubQueryResponseResult;
```

```
}  
  
}
```

### 4.4.3 分页与高亮

```
@Service  
public class EsCourseService {  
    private static final Logger LOGGER = LoggerFactory.getLogger(EsCourseService.class);  
  
    @Value("${xuecheng.elasticsearch.course.index}")  
    private String es_index;  
    @Value("${xuecheng.elasticsearch.course.type}")  
    private String es_type;  
    @Value("${xuecheng.elasticsearch.course.source_field}")  
    private String source_field;  
  
    @Autowired  
    RestHighLevelClient restHighLevelClient;  
  
    public QueryResponseResult<CoursePub> list(int page,int size,CourseSearchParam  
courseSearchParam) {  
  
        //设置索引  
        SearchRequest searchRequest = new SearchRequest(es_index);  
        //设置类型  
        searchRequest.types(es_type);  
        SearchSourceBuilder searchSourceBuilder = new SearchSourceBuilder();  
        BoolQueryBuilder boolQueryBuilder = QueryBuilders.boolQuery();  
        //source源字段过滤  
        String[] source_fields = source_field.split(",");  
        searchSourceBuilder.fetchSource(source_fields, new String[]{});  
        //关键字  
        if(StringUtils.isEmpty(courseSearchParam.getKeyword())){  
            //匹配关键字  
            MultiMatchQueryBuilder multiMatchQueryBuilder =  
                QueryBuilders.multiMatchQuery(courseSearchParam.getKeyword(), "name",  
                "teachplan","description");  
            //设置匹配占比  
            multiMatchQueryBuilder.minimumShouldMatch("70%");  
            //提升另一个字段的Boost值  
            multiMatchQueryBuilder.field("name",10);  
            boolQueryBuilder.must(multiMatchQueryBuilder);  
        }  
        //过滤  
        if(StringUtils.isEmpty(courseSearchParam.getMt())){  
            boolQueryBuilder.filter(QueryBuilders.termQuery("mt",courseSearchParam.getMt()));  
        }  
        if(StringUtils.isEmpty(courseSearchParam.getSt())){
```



```
        boolQueryBuilder.filter(QueryBuilders.termQuery("st",courseSearchParam.getSt()));
    }
    if(StringUtils.isEmpty(courseSearchParam.getGrade())){

boolQueryBuilder.filter(QueryBuilders.termQuery("grade",courseSearchParam.getGrade()));
    }
    //分页
    if(page<=0){
        page = 1;
    }
    if(size<=0){
        size = 20;
    }
    int start = (page-1)*size;
    searchSourceBuilder.from(start);
    searchSourceBuilder.size(size);
    //布尔查询
    searchSourceBuilder.query(boolQueryBuilder);
    //高亮设置
    HighlightBuilder highlightBuilder = new HighlightBuilder();
    highlightBuilder.preTags("<font class='eslight'>");
    highlightBuilder.postTags("</font>");
    //设置高亮字段
    highlightBuilder.fields().add(new HighlightBuilder.Field("name"));
    searchSourceBuilder.highlighter(highlightBuilder);
    //请求搜索
    searchRequest.source(searchSourceBuilder);
    SearchResponse searchResponse = null;
    try {
        searchResponse = restHighLevelClient.search(searchRequest);
    } catch (IOException e) {
        e.printStackTrace();
        LOGGER.error("xuecheng search error..{}",e.getMessage());
        return new QueryResponseResult(CommonCode.SUCCESS,new QueryResult<CoursePub>());
    }

    //结果集处理
    SearchHits hits = searchResponse.getHits();
    SearchHit[] searchHits = hits.getHits();
    //记录总数
    long totalHits = hits.getTotalHits();
    //数据列表
    List<CoursePub> list = new ArrayList<>();

    for (SearchHit hit : searchHits) {
        CoursePub coursePub = new CoursePub();

        //取出source
        Map<String, Object> sourceAsMap = hit.getSourceAsMap();

        //取出名称
        String name = (String) sourceAsMap.get("name");

        //取出高亮字段内容
```



```
Map<String, HighlightField> highlightFields = hit.getHighlightFields();
if(highlightFields!=null){
    HighlightField nameField = highlightFields.get("name");
    if(nameField!=null){
        Text[] fragments = nameField.getFragments();
        StringBuffer stringBuffer = new StringBuffer();
        for(Text str : fragments) {
            stringBuffer.append(str.string());
        }
        name = stringBuffer.toString();
    }
}
coursePub.setName(name);
//图片
String pic = (String) sourceAsMap.get("pic");
coursePub.setPic(pic);
//价格
Float price = null;
try {
    if(sourceAsMap.get("price")!=null ){
        price = Float.parseFloat((String) sourceAsMap.get("price"));
    }

} catch (Exception e) {
    e.printStackTrace();
}
coursePub.setPrice(price);
Float price_old = null;
try {
    if(sourceAsMap.get("price_old")!=null ){
        price_old = Float.parseFloat((String) sourceAsMap.get("price_old"));
    }
} catch (Exception e) {
    e.printStackTrace();
}
coursePub.setPrice_old(price_old);

list.add(coursePub);

}
QueryResult<CoursePub> queryResult = new QueryResult<>();
queryResult.setList(list);
queryResult.setTotal(totalHits);
QueryResponseResult<CoursePub> coursePubQueryResponseResult = new
QueryResponseResult<CoursePub>(CommonCode.SUCCESS,queryResult);
return coursePubQueryResponseResult;
}
}
```



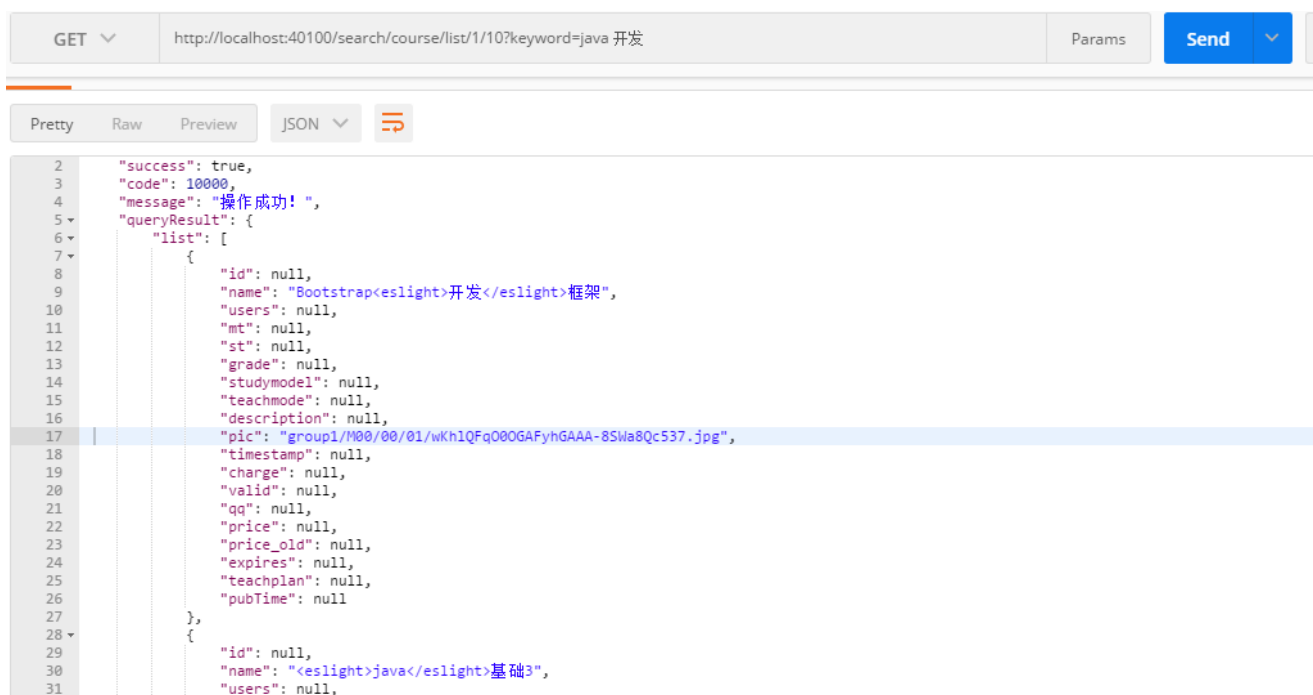
## 4.5 Controller

```
@RestController
@RequestMapping("/search/course")
public class EsCourseController implements EsCourseControllerApi {
    @Autowired
    EsCourseService esCourseService;

    @Override
    @GetMapping(value="/list/{page}/{size}")
    public QueryResponseResult<CoursePub> list(@PathVariable("page") int page,
    @PathVariable("size") int size, CourseSearchParam courseSearchParam) throws IOException {
        return esCourseService.list(page,size,courseSearchParam);
    }
}
```

## 4.5 测试

使用postman测试/search/course



The screenshot shows a Postman interface with a GET request to `http://localhost:40100/search/course/list/1/10?keyword=java 开发`. The response is in JSON format, showing a successful operation with a list of course results.

```
{
  "success": true,
  "code": 10000,
  "message": "操作成功!",
  "queryResult": {
    "list": [
      {
        "id": null,
        "name": "Bootstrap<eslight>开发</eslight>框架",
        "users": null,
        "mt": null,
        "st": null,
        "grade": null,
        "studymodel": null,
        "teachmode": null,
        "description": null,
        "pic": "group1/M00/00/01/wKh1QFq00OGAFyhGAAA-8SWa8Qc537.jpg",
        "timestamp": null,
        "charge": null,
        "valid": null,
        "qq": null,
        "price": null,
        "price_old": null,
        "expires": null,
        "teachplan": null,
        "pubTime": null
      },
      {
        "id": null,
        "name": "<eslight>java</eslight>基础3",
        "users": null,

```