



链滴

集成 solr 复盘

作者: [luomuren](#)

原文链接: <https://ld246.com/article/1661181539788>

来源网站: [链滴](#)

许可协议: [署名-相同方式共享 4.0 国际 \(CC BY-SA 4.0\)](#)

公司有个平台要在微服务中集成搜索引擎，选了solr，第一次研究搜索引擎，记录一下集成过程
在csdn上找了一篇教程：[springboot集成solr实现全局搜索系列](#)，作者写的很详细

环境

jdk: jdk1.8.0_51

Solr: solr-8.11.2

tomcat: tomcat 8

微服务框架: ruoyi-cloud-2.4.0

项目相关文件git地址: <https://gitee.com/lmr-replay/replay-solr>

安装并配置solr

这里基本都是参照教程安装的，不会安装的小伙伴可以看一下上面的教程，很详细

springboot集成solr

这块我选择作为一个单独的服务集成到微服务中，方便后期其他项目引入

因为跟数据库做的定时同步，这块只实现了查询方法

core: article_core、business_core、recruit_core、course_core

每个core第一的字段都相同: id、title、content、author、keywords

1. 首先引入mave

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-solr</artifactId>
</dependency>
```

2. 在nacos中配置solr服务端地址

```
spring:
  data:
    solr:
      host: http://localhost:8090/solr/
```

3. 创建solr获取的数据对象实体类

项目这块的需求是要同时查询多个表的数据，表之间没有关联，这块想了好久，不知道是要定义一个core来处理还是应该创建多个core来处理

首先配置了一个core来处理多个表的数据，有很多问题，处理起来太麻烦，这块因为催的比较紧急，择了每个表创建一个core

这块有大神知道这个问题怎么解决的，有时间的话麻烦私聊指教一下，
raypraypray

- 创建单个Core对象：CoreParam

```
public class CoreParam {
    /** 查询的core */
    private String core;
    /** 默认查询字段 */
    private String dfFiled;
    /** 排序 (id desc,name asc) */
    private String sort;

    public String getCore() {
        return core;
    }

    public void setCore(String core) {
        this.core = core;
    }

    public String getDfFiled() {
        return dfFiled;
    }

    public void setDfFiled(String dfFiled) {
        this.dfFiled = dfFiled;
    }

    public String getSort() {
        return sort;
    }

    public void setSort(String sort) {
        this.sort = sort;
    }
}
```

- 创建Solr查询参数对象：SolrSearchParam

```
public class SolrSearchParam {
    /** 索引 */
    private String id;

    /** 用户输入的查询字符串 */
    private String q;

    /** 分页定义结果起始记录数 */
    private Integer start;

    /** 分页定义结果每页返回记录数 */
    private Integer rows;

    /**
     * 需要检索的core列表
     */
    private List<CoreParam> coreParams;
```

```

public String getId() {
    return id;
}

public void setId(String id) {
    this.id = id;
}

public String getQ() {
    return q;
}

public void setQ(String q) {
    this.q = q;
}

public Integer getStart() {
    return start;
}

public void setStart(Integer start) {
    this.start = start;
}

public Integer getRows() {
    return rows;
}

public void setRows(Integer rows) {
    this.rows = rows;
}

public List<CoreParam> getCoreParams() {
    return coreParams;
}

public void setCoreParams(List<CoreParam> coreParams) {
    this.coreParams = coreParams;
}
}

```

- Solr分页数据返回类：SolrDataInfo

```

public class SolrDataInfo implements Serializable {
    private static final long serialVersionUID = 1L;
    private long total;
    private List<?> rows;
    private int code;
    private String msg;
    private String title; // 标题
    private String authorTitle; // 作者字段标题

    public String getAuthorTitle() {
        return authorTitle;
    }
}

```

```

public void setAuthorTitle(String authorTitle) {
    this.authorTitle = authorTitle;
}

public long getTotal() {
    return total;
}

public void setTotal(long total) {
    this.total = total;
}

public List<?> getRows() {
    return rows;
}

public void setRows(List<?> rows) {
    this.rows = rows;
}

public int getCode() {
    return code;
}

public void setCode(int code) {
    this.code = code;
}

public String getMsg() {
    return msg;
}

public void setMsg(String msg) {
    this.msg = msg;
}

public String getTitle() {
    return title;
}

public void setTitle(String title) {
    this.title = title;
}
}

```

4. 创建SolrCore枚举

定义页面上显示的数据类型、参数名称及详情页url

```

/**
 * SolrCore枚举
 */
public enum SolrCoreEnum {
    ARTICLE_CORE("article_core", "/articleDetail?id=#{id}", "资讯", "作者"),

```

```

BUSINESS_CORE("business_core", "/companyDetail/?id=#{id}", "企业", "行业"),
RECRUIT_CORE("recruit_core", "/recruitDetail?id=#{id}", "招聘信息", ""),
COURSE_CORE("course_core", "/courseDetail/?courseId=#{id}", "就业课程", "课程类型");
private String core; // Solr Core
private String detailUrl; // 详情页链接（使用#{id}注明id拼接位置）
private String title; // 标题
private String authorTitle; // 作者字段标题

public static String getDetailUrl(String core, Long id) {
    for (SolrCoreEnum solrCore : SolrCoreEnum.values()) {
        if (solrCore.getCore().equals(core)) {
            if (StringUtils.isNotBlank(id)) {
                solrCore.detailUrl = solrCore.detailUrl.replaceAll("\\#{id\\}", String.valueOf(id));
            }
            return solrCore.detailUrl;
        }
    }
    return "";
}

public static String getTitle(String core) {
    for (SolrCoreEnum solrCore : SolrCoreEnum.values()) {
        if (solrCore.getCore().equals(core)) {
            return solrCore.title;
        }
    }
    return "";
}

public static String getAuthorTitle(String core) {
    for (SolrCoreEnum solrCore : SolrCoreEnum.values()) {
        if (solrCore.getCore().equals(core)) {
            return solrCore.authorTitle;
        }
    }
    return "";
}

public String getCore() {
    return core;
}

SolrCoreEnum() {
}

SolrCoreEnum(String core, String detailUrl, String title, String authorTitle) {
    this.core = core;
    this.detailUrl = detailUrl;
    this.title = title;
    this.authorTitle = authorTitle;
}
}

```

5. 创建Solr查询Service

```

/**
 * @ClassName SolrSearchParam
 * @Author lmr
 * <p>
 * SolrService
 * <p>
 * @Date 2022/8/12 0012 16:34
 * @Version 1.0
 */
public interface SolrService {
    /**
     * 查询
     * @param param
     * @return
     * @throws IOException
     * @throws SolrServerException
     */
    AjaxResult querySolr(SolrSearchParam param) throws IOException, SolrServerException;
}

```

6. 创建Solr查询Service实现

```

@Service
public class SolrServiceImpl implements SolrService {
    @Autowired
    private SolrClient client;
    /**
     * 高亮查询
     */
    @Override
    public AjaxResult querySolr(SolrSearchParam param) throws IOException, SolrServerException {
        HashMap<String, Object> map = new HashMap<>();
        // 获取所有需要查询的core
        List<CoreParam> coreParams = param.getCoreParams();
        // 查询到的记录总行数
        Long total = 0L;
        for (CoreParam core : coreParams) {
            SolrDocumentList solrDocuments;
            if (StringUtils.isNull(param.getStart())) {
                solrDocuments = querySolr(core.getCore(), param.getQ(), core.getDfFiled(), core.getSort());
            } else {
                solrDocuments = querySolr(core.getCore(), param.getQ(), core.getDfFiled(), core.getSort(), param.getStart(), param.getRows());
            }
            for (SolrDocument solrDocument : solrDocuments) {
                // 从枚举中获取详情页url并拼装id
                solrDocument.setField("detailUrl", SolrCoreEnum.getDetailUrl(core.getCore(), Long.parseLong(solrDocument.getFieldValue("id").toString())));
            }
            // 组装分页数据
            SolrDataInfo rspData = new SolrDataInfo();
            rspData.setRows(solrDocuments);
            rspData.setMsg("查询成功");
        }
    }
}

```

```

        long numFound = solrDocuments.getNumFound();
        total += numFound;
        // 单类型记录数
        rspData.setTotal(numFound);
        rspData.setTitle(SolrCoreEnum.getTitle(core.getCore()));
        rspData.setAuthorTitle(SolrCoreEnum.getAuthorTitle(core.getCore()));
        // 以core作为key返回数据
        map.put(core.getCore(), rspData);
    }
    map.put("total", total);
    return AjaxResult.success(map);
}

/**
 * 查询
 * @param core 要查询的core
 * @param q 用户输入的字符串
 * @param dfField 默认的查询字段
 * @param sort 排序字段
 * @return
 * @throws IOException
 * @throws SolrServerException
 */
public SolrDocumentList querySolr(String core, String q, String dfField, String sort) throws
IOException, SolrServerException {
    return querySolr(core, q, dfField, sort, 0, 10);
}

/**
 * 查询
 * @param core 要查询的core
 * @param q 用户输入的字符串
 * @param dfField 默认的查询字段
 * @param sort 排序字段
 * @param start 分页定义结果起始记录数
 * @param rows 分页定义结果每页返回记录数
 * @return
 * @throws IOException
 * @throws SolrServerException
 */
public SolrDocumentList querySolr(String core, String q, String dfField, String sort, Integer s
art, Integer rows) throws IOException, SolrServerException {
    SolrQuery params = new SolrQuery();
    //查询条件, 这里的 q 对应 下面图片标红的地方
    params.set("q", q);
    // 过滤条件
    // params.set("fq", "product_price:[100 TO 100000]");
    // 组装排序
    if (!StringUtils.isEmpty(sort)) {
        // 多个字段使用,分开
        String[] sorts = sort.split(",");
        for (String sortItem : sorts) {
            // 字段和排序方式使用' '分开
            String[] items = sortItem.split(" ");

```

```

        if (items.length > 1) {
            params.addSort(new SolrQuery.SortClause(items[0], items[1]));
        } else {
            params.addSort(new SolrQuery.SortClause(items[0], SolrQuery.ORDER.asc));
        }
    }
}
//分页
params.setStart(start);
params.setRows(rows);
//默认域
params.set("df", dfField);
//只查询指定域
//    params.set("fl", content + ",content,id,title,author");
//高亮
//打开开关
params.setHighlight(true);
//指定高亮域
params.addHighlightField("id");
params.addHighlightField(dfField);
params.addHighlightField("title");
params.addHighlightField("content");
params.addHighlightField("author");
params.addHighlightField("keywords");

//设置前缀
params.setHighlightSimplePre("<span style='color:red'>");
//设置后缀
params.setHighlightSimplePost("</span>");

QueryResponse queryResponse = client.query(core,params);

SolrDocumentList results = queryResponse.getResults();

//获取高亮显示的结果, 高亮显示的结果和查询结果是分开放的
Map<String, Map<String, List<String>>> highlight = queryResponse.getHighlighting();

for(SolrDocument result : results){ // 将高亮结果合并到查询结果中
    highlight.forEach((k,v) ->{
        if(result.get("id").equals(k)){
            v.forEach((k1,v1) -> {
                result.setField(k1,v1); // 高亮列合并如结果
            });
        }
    });
}
return results;
}
}

```

7. Solr查询Controller

****getByBgImage()****方法主要用于获取必应壁纸

```

@RestController
@RequestMapping("/solr")
public class SolrController {
    @Autowired
    private SolrClient client;
    @Autowired
    private SolrService solrService;

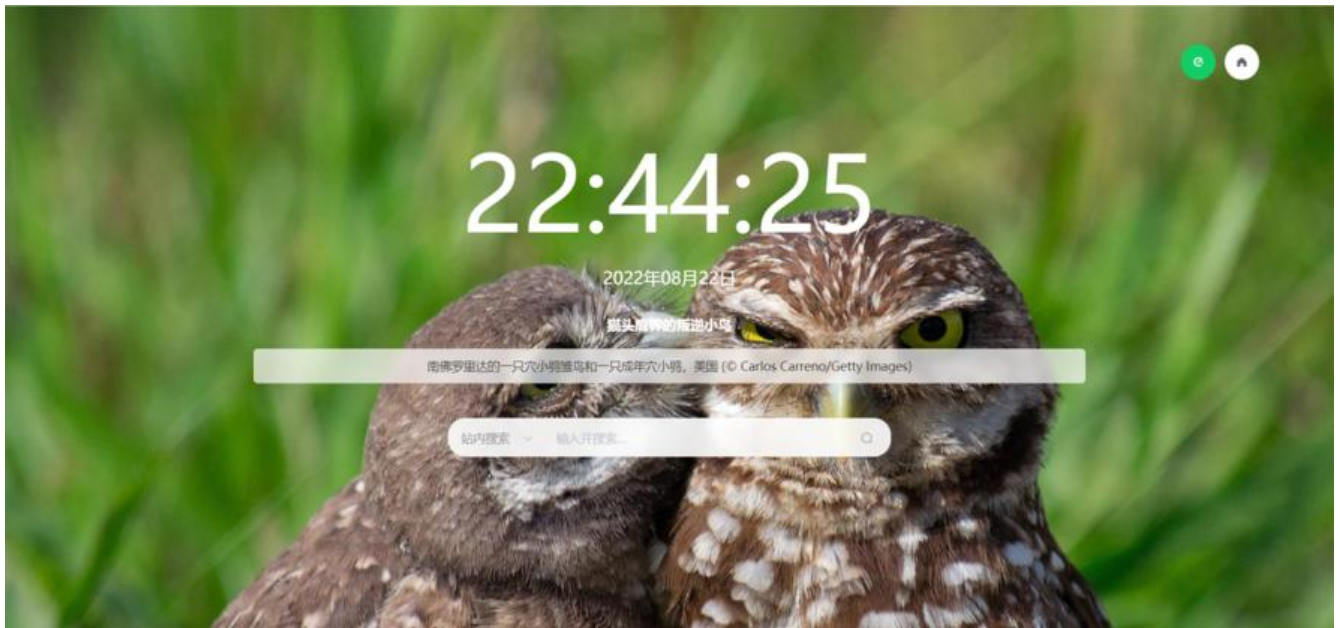
    /**
     * 按条件查询, 条件过滤, 排序, 分页, 高亮显示, 获取部分域信息
     * @return
     */
    @PostMapping("/search")
    public AjaxResult search(@RequestBody SolrSearchParam param){
        if (CollectionUtils.isEmpty(param.getCoreParams())) {
            return AjaxResult.error("请指定core");
        }
        try {
            return solrService.querySolr(param);
        } catch (IOException e) {
        } catch (SolrServerException e) {
        }
        return AjaxResult.error("查询失败, 请联系服务员");
    }

    /**
     * 获取每日壁纸 (必应壁纸)
     * @param
     * @return
     */
    @GetMapping("/getBgImageByBy/{idx}")
    public AjaxResult getByBgImage(@PathVariable String idx){
        try {
            // 通过传入1-10的随机idx获取1-10天随机一天的壁纸
            String s = HttpUtils.sendGet("http://cn.bing.com/HPIImageArchive.aspx?format=js&n1&idx=" + idx);
            JSONObject jsonObject = JSON.parseObject(s);
            return AjaxResult.success(jsonObject);
        } catch (Exception e) {
            return AjaxResult.success("壁纸获取失败");
        }
    }
}

```

前端搜索页面

在ruoyi微服务的ui中仿照csdn谷歌浏览器插件首页自己写了一个搜索页面页面



联系方式

作者: [永夜](#)

邮箱: Evernight@aliyun.com

以上内容有不正确的地方烦请指正! ☹️pray☹️
ray☹️pray