复制从 索引同步接口文档

需要依赖sdk

<dependency>

<groupId>com.peter.search</groupId>

<artifactId>search-api</artifactId>

<version>参见版本号说明页</version>

</dependency>

一、增量索引同步

描述：当数据放生变化时（修改/删除），需要将变化更新到索引中，现有三种增量同步方式可供使用，介绍如下

1、接入方式一：调用接口同步

```
deltaImportData(UpdateRequestDTO updateRequest)
```

接口入参 UpdateRequestDTO

参数名	参数类型	是否必须	
serviceTag	string	是	
opType	enum	是	INSERT：如果文档不存在，则insert,如果文档已经存在，则删除原有文档，再insert UPSERT：如果文档不存在，则insert,如果文档已经存在，则在原有文档基础上做更新,UPSERT不如INSERT性能好 UPDATE：如果文档不存在，则不做任何操作,如果文档已经存在，则在原有文档基础上做更新 DELETE：删除
docDataList	List<DocData>	是	

DocData类型

参数名	参数类型	是否必须	

docId	string	是	文档ID
routing	string	否	路由，用于查询时使用
docData	string	是	要同步的数据
docDataType	string	是	目前只支持json

接口返回参数

参数名	参数类型	是否必须	描述
status	int	是	1：成功，2：失败
errMsg	string	否	错误信息描述


```
{
  "status":1,
  "errMsg":""
}
```


2、接入方式二：增量同步消息

group name:index_data_producer_group

topic name:index_data_topic

MQ消息格式

tags			
tags	serviceTag		
keys	OP_TYPE		
body	json	`JSON.toJSONString(docDataList)`	参见上方List<DocData>数据类型定义

首先：增加如下类：

configuration

```
@Configuration
public class SyncDataMQClientConfig {

    @Value("${rocketmq.producer.sendMsgTimeout}")
    private Integer timeout;

    @Value("${rocketmq.producer.namesrvAddr}")
    private String address;

    @Value("${rocketmq.producer.maxMessageSize}")
    private Integer maxMessageSize;

    @Value("${rocketmq.producer.retryTimesWhenSendFailed}")
    private Integer retryTimes;

    @Bean
    public SyncDataMQClient syncDataMQClient(){
        SyncDataMQClient client = new SyncDataMQClient();
        client.setAddress(address);
        client.setMaxMessageSize(maxMessageSize);
        client.setRetryTimes(retryTimes);
        client.setTimeout(timeout);
        return client;
    }
}
```

使用sdk的syncDataMQClient，代码示例：

```
    @Autowired
    SyncDataMQClient syncDataMQClient;

    public void brandSync(List<Brand> brandList) {

        try {
            List<DocData> docDataList = sesDataBuilder.buildBrandData(brandList);
            syncDataMQClient.syncData(SystemCore.BRAND.getName(), OP_TYPE.UPSERT, docDataList);
        } catch (Exception e) {
            log.error(e.getMessage(), e);
        }
    }
```

3、接入方式三：使用注解

@SyncDataPoint使用示例

注解会从return出来的对象解析数据，返回对象支持以下数据类型

DTO对象

list<DTO对象>

Map

List<Map>

String、Integer、Integer （仅限于OP_TYOE=DELETE删除操作时）

List<String>、List<Integer>、List<Integer>（仅限于OP_TYOE=DELETE删除操作时）

```
@SyncDataPoint(serviceTag="product", docIdParamName="id")
public ProductDTO updateProduct(Long id, String price, int sizeChart, String sales){
        ProductDTO product = new ProductDTO();
        product.setId(id);
        product.setPrice(price);
        product.setSizeChart(sizeChart);
        product.setSales(sales);
        daoService.updateProduct(product);
        return product;
}
```

1com.peter.search.annotation

2MQ

```
###producer
rocketmq.producer.groupName=${spring.application.name}
#mqnameserver
rocketmq.producer.namesrvAddr=
# 1024*4(4M)
rocketmq.producer.maxMessageSize=4096
#,3000
rocketmq.producer.sendMsgTimeout=3000
#2
rocketmq.producer.retryTimesWhenSendFailed=2
```

三、全量索引同步

描述：由于数据结构修改，或者错误数据修复等问题，索引需要定时或手动进行数据全量同步。所以各数据提供方应该提供分页查询接口，供search-service

调用，接口文档如下

1、入参：

参数名	是否必须	参数类型	描述

currentPage	是	int	取第几页数据
pageSize	是	int	每页返回多少条数据

2、返回参数

一级参数名	二级参数名	是否必须	参数类型	描述
status			int	1-成功 0-失败
page	totalCount	是	int	总共多少条
	totalPage	是	int	总页数
	currentPage	是	int	当前页
	isLastPage	是	boolean	是否最后一页
dataList	docId	是	number	数据的唯一ID
	routing	否	string	路由值，如果自己查询时需要路由，则该值不能为空
	docData	是	json	返回的数据

3、返回参数示例


```
{
    “status”:1,
  "page":{
     "totalCount":500,
     "totalPage":5,
     "currentPage":1,
     "isLastPage":0
  },
  "dataList":[
```

```
    {
      "docId":1,
      "routing":123456,
      "docData":{
        "id":1,
        "name":"王",
        "age":2,
        "weight":60
      }
    },

    {
      "docId":2,
      "routing":123456,
      "docData":{
        "id":2,
        "name":"王",
        "age":2,
        "weight":60
      }
    }]
}
```