open search示例代码

```
package com.granwin.customer.util.aws.search;

import com.alibaba.fastjson.JSON;
import com.alibaba.fastjson.JSONObject;
import com.granwin.common._enum.ApiResultEnum;
import com.granwin.common._enum.OpenSearchResultEnum;
import com.granwin.common.entity.opensearch.Problem;
import com.granwin.common.response.DefaultResponse;
import com.granwin.common.response.FailResponse;
import com.granwin.common.response.Response;
import org.apache.http.HttpHost;
import org.apache.http.auth.AuthScope;
import org.apache.http.auth.UsernamePasswordCredentials;
import org.apache.http.client.CredentialsProvider;
import org.apache.http.impl.client.BasicCredentialsProvider;
import org.opensearch.client.RestClient;
import org.opensearch.client.json.jackson.JacksonJsonpMapper;
import org.opensearch.client.opensearch.OpenSearchClient;
import org.opensearch.client.opensearch._types.OpenSearchException;
import org.opensearch.client.opensearch.core.*;
import org.opensearch.client.opensearch.indices.*;
import org.opensearch.client.transport.rest_client.RestClientTransport;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;

import java.io.IOException;
import java.lang.reflect.ParameterizedType;
import java.lang.reflect.Type;
import java.util.Objects;



@Component("com.granwin.customer.util.aws.search.OpenSearchUtil")
public class OpenSearchUtil {
    private Logger log = LoggerFactory.getLogger(getClass());

    @Value("${open-search.url}")
    private String url;

    @Value("${open-search.username}")
    private String username;

    @Value("${open-search.password}")
    private String password;

    @Value("${open-search.port}")
    private Integer port;

    private OpenSearchClient openSearchClient;
```

```
    {
        try {
            final CredentialsProvider credentialsProvider = new
BasicCredentialsProvider();
            credentialsProvider.setCredentials(AuthScope.ANY,
                    new UsernamePasswordCredentials("granwin", "Granwin888."));

            RestClient restClient = RestClient.builder(new HttpHost("search-
granwin-es-s2i6fyqu46o2uabv4y3rwcsqym.us-west-2.es.amazonaws.com",443,"https")).
                    setHttpClientConfigCallback(httpClientBuilder ->
httpClientBuilder.setDefaultCredentialsProvider(credentialsProvider)).build();
            RestClientTransport restClientTransport = new
RestClientTransport(restClient, new JacksonJsonpMapper());
            this.openSearchClient = new OpenSearchClient(restClientTransport);
        }catch (Exception e){
         log.error("OpenSearchUtil初始化异常");
        }
    }

    /**
     * 添加索引
     * @param index
     * @return
     */
    public Response creatIndex(String index){
        CreateIndexRequest createIndexRequest = new
CreateIndexRequest.Builder().index(index).build();
        try {
            CreateIndexResponse createIndexResponse =
openSearchClient.indices().create(createIndexRequest);
        } catch (Exception e) {
            log.error("创建索引失败");
            e.printStackTrace();
            return new FailResponse(ApiResultEnum.FAIL,"创建索引失败");
        }
        return new DefaultResponse();
    }

    /**
     * 设置索引自动扩展的副本数量
     * @param index
     * @param replicas 副本数量  “0-all”
     * @return
     */
    public Response setIndex(String index,String replicas){
        IndexSettings indexSettings = new
IndexSettings.Builder().autoExpandReplicas(replicas).build();
        PutIndicesSettingsRequest putIndicesSettingsRequest = new
PutIndicesSettingsRequest.Builder().index(index).settings(indexSettings).build()
;
        try {
            PutIndicesSettingsResponse putIndicesSettingsResponse =
openSearchClient.indices().putSettings(putIndicesSettingsRequest);
            System.out.println(JSON.toJSONString(putIndicesSettingsResponse));
        } catch (Exception e) {
            log.error("设置索引失败");
            return new FailResponse(ApiResultEnum.FAIL,"设置索引失败");
        }
```

```
        return new DefaultResponse();
    }

    /**
     * 向索引添加数据
     * @param data  添加的数据
     * @param index 索引
     * @return
     */
    public Response addData(Object data,String index){
        IndexRequest<Object> indexRequest = new IndexRequest.Builder<>
().index(index).document(data).build();
        try {
            IndexResponse indexResponse = openSearchClient.index(indexRequest);
            if (!Objects.equals(indexResponse.result().jsonValue(),
OpenSearchResultEnum.create.getName())){
                return new FailResponse(ApiResultEnum.FAIL,"添加数据失败");
            }
        }catch (OpenSearchException e){
            //添加索引
            Response response = creatIndex(index);
            if (response.getCode() != ApiResultEnum.SUCCESS.getId()){
                return new FailResponse(ApiResultEnum.FAIL,"添加数据失败,索引创建失
败");
            }
            try {
                IndexResponse indexResponse =
openSearchClient.index(indexRequest);
            } catch (IOException ex) {
                log.error("添加数据失败");
                return new FailResponse(ApiResultEnum.FAIL,"添加数据失败");
            }
        }
        catch (Exception e) {
            log.error("添加数据失败");
            return new FailResponse(ApiResultEnum.FAIL,"添加数据失败");
        }
        return new DefaultResponse();
    }

    /**
     * 根据数据id进行更新
     * @param data 数据
     * @param id 数据id
     * @return
     */
    public Response updateData(Object data,String id,String index){
        UpdateRequest<Object, Object> updateRequest = new
UpdateRequest.Builder<>().index(index).id(id).doc(data).build();
        try {
            UpdateResponse<Object> updateResponse =
openSearchClient.update(updateRequest, Object.class);
            if (!Objects.equals(updateResponse.result().jsonValue(),
OpenSearchResultEnum.update.getName())){
                return new FailResponse(ApiResultEnum.FAIL,"更新数据失败");
            }
        } catch (Exception e) {
            log.error("更新数据失败,index={}  id={}",index,id);
```

```
            return new FailResponse(ApiResultEnum.FAIL,"更新数据失败");
        }
        return new DefaultResponse();
    }

    /**
     * 根据数据id和索引进行删除
     * @param index
     * @param id
     * @return
     */
    public Response deleteData(String index,String id){
        try {
            DeleteResponse deleteResponse = openSearchClient.delete(b ->
b.index(index).id(id));
            if (!Objects.equals(deleteResponse.result().jsonValue(),
OpenSearchResultEnum.delete.getName())){
                return new FailResponse(ApiResultEnum.FAIL,"删除数据失败");
            }
        } catch (Exception e) {
            log.error("删除数据失败,index={}  id={}",index,id);
            return new FailResponse(ApiResultEnum.FAIL,"更新数据失败");
        }
        return new DefaultResponse();
    }

    /**
     * 删除索引
     * @param index
     * @return
     */
    public Response deleteIndex(String index){
        DeleteIndexRequest deleteRequest = new
DeleteIndexRequest.Builder().index(index).build();
        try {
            DeleteIndexResponse deleteIndexResponse =
openSearchClient.indices().delete(deleteRequest);
        } catch (Exception e) {
            log.error("删除索引失败,index={}",index);
            return new FailResponse(ApiResultEnum.FAIL,"删除索引失败");
        }
        return new DefaultResponse();
    }

    public OpenSearchClient getOpenSearchClient(){
        return openSearchClient;
    }

    /*排序
    SearchResponse<SmartReply> searchResponse = openSearchClient.search(s ->
s.index(INDEX).sort(sort->sort.field(field-
>field.field("count").order(SortOrder.Desc))), SmartReply.class);*/
    /*
     /*分页
            int pageSize = 1;
            int pageNum= 1;
```

```
            SearchResponse<SmartReply> searchResponse =
openSearchClient.search(s -> s.index("smart-
reply").from(pageNum*pageSize).size(pageSize), SmartReply.class);*/
}
```