package com.twitter.search.earlybird.document;

import java.io.IOException;

import javax.annotation.Nullable;

import org.apache.commons.codec.binary.Base64;

import org.apache.lucene.document.Document;

import org.apache.lucene.document.Field;

import org.apache.lucene.document.FieldType;

import org.apache.lucene.index.IndexableField;

import org.apache.thrift.TBase;

import org.apache.thrift.TException;

import org.apache.thrift.TSerializer;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import com.twitter.search.common.metrics.SearchCounter;

import com.twitter.search.common.util.text.OmitNormTextField;

import com.twitter.search.earlybird.exception.CriticalExceptionHandler;

/\*\*

\* Factory that constructs a Lucene document from a thrift object stored in T format.

\*

\* @param <T> ThriftStatus or ThriftIndexingEvent, to be converted to a Lucene Document.

\*/

public abstract class DocumentFactory<T extends TBase<T, ?>> {

private static final Logger LOG = LoggerFactory.getLogger(DocumentFactory.class);

private static final int MAX\_ALLOWED\_INVALID\_DOCUMENTS = 100;

private static final SearchCounter INVALID\_DOCUMENTS\_COUNTER =

SearchCounter.export("invalid\_documents");

private final CriticalExceptionHandler criticalExceptionHandler;

public DocumentFactory(CriticalExceptionHandler criticalExceptionHandler) {

this.criticalExceptionHandler = criticalExceptionHandler;

}

/\*\*

\* Given the thrift representation of a tweet, returns the associated tweetId.

\*/

public abstract long getStatusId(T thriftObject);

/\*\*

\* Given the thrift representation of a tweet, returns a Lucene Document with all the fields

\* that need to be indexed.

\*/

@Nullable

public final Document newDocument(T thriftObject) {

try {

return innerNewDocument(thriftObject);

} catch (Exception e) {

String statusId = "Not available";

if (thriftObject != null) {

try {

statusId = Long.toString(getStatusId(thriftObject));

} catch (Exception ex) {

LOG.error("Unable to get tweet id for document", ex);

statusId = "Not parsable";

}

}

LOG.error("Unexpected exception while indexing. Status id: " + statusId, e);

if (thriftObject != null) {

// Log the status in base64 for debugging

try {

LOG.warn("Bad ThriftStatus. Id: " + statusId + " base 64: "

+ Base64.encodeBase64String(new TSerializer().serialize(thriftObject)));

} catch (TException e1) {

// Ignored since this is logging for debugging.

}

}

INVALID\_DOCUMENTS\_COUNTER.increment();

if (INVALID\_DOCUMENTS\_COUNTER.get() > MAX\_ALLOWED\_INVALID\_DOCUMENTS) {

criticalExceptionHandler.handle(this, e);

}

return new Document();

}

}

/\*\*

\* Given the thrift representation of a tweet, returns a Lucene Document with all the fields

\* that need to be indexed.

\*

\* Return null if the given thrift object is invalid.

\*

\* @throws IOException if there are problems reading the input of producing the output. Exception

\* is handled in {@link #newDocument(TBase)}.

\*/

@Nullable

protected abstract Document innerNewDocument(T thriftObject) throws IOException;

// Helper methods that prevent us from adding null fields to the lucene index

protected void addField(Document document, IndexableField field) {

if (field != null) {

document.add(field);

}

}

protected Field newField(String data, String fieldName) {

return newField(data, fieldName, OmitNormTextField.TYPE\_NOT\_STORED);

}

protected Field newField(String data, String fieldName, FieldType fieldType) {

if (data != null) {

return new Field(fieldName, data, fieldType);

}

return null;

}

}