package com.twitter.search.common.schema;

import java.io.IOException;

import java.util.List;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.annotation.Nullable;

import com.twitter.common.text.util.PositionIncrementAttributeSerializer;

import com.twitter.common.text.util.TokenStreamSerializer;

import com.twitter.search.common.schema.base.FieldNameToIdMapping;

import com.twitter.search.common.schema.thriftjava.ThriftDocument;

import com.twitter.search.common.schema.thriftjava.ThriftField;

import com.twitter.search.common.schema.thriftjava.ThriftFieldData;

import com.twitter.search.common.schema.thriftjava.ThriftGeoCoordinate;

import com.twitter.search.common.util.analysis.CharTermAttributeSerializer;

import com.twitter.search.common.util.analysis.LongTermAttributeSerializer;

import com.twitter.search.common.util.analysis.LongTermsTokenStream;

import com.twitter.search.common.util.analysis.PayloadAttributeSerializer;

import com.twitter.search.common.util.analysis.PayloadWeightedTokenizer;

import com.twitter.search.common.util.spatial.GeoUtil;

/\*\*

\* Builder class for building ThriftDocuments.

\*/

public class ThriftDocumentBuilder {

private static final Logger LOG = Logger.getLogger(ThriftDocumentBuilder.class.getName());

protected final ThriftDocument doc = new ThriftDocument();

protected final FieldNameToIdMapping idMapping;

private static final ThreadLocal<TokenStreamSerializer> PAYLOAD\_WEIGHTED\_SERIALIZER\_PER\_THREAD =

new ThreadLocal<TokenStreamSerializer>() {

@Override

protected TokenStreamSerializer initialValue() {

return TokenStreamSerializer.builder()

.add(new CharTermAttributeSerializer())

.add(new PositionIncrementAttributeSerializer())

.add(new PayloadAttributeSerializer())

.build();

}

};

private static final ThreadLocal<TokenStreamSerializer> LONG\_TERM\_SERIALIZER\_PER\_THREAD =

new ThreadLocal<TokenStreamSerializer>() {

@Override

protected TokenStreamSerializer initialValue() {

return TokenStreamSerializer.builder()

.add(new LongTermAttributeSerializer())

.build();

}

};

public ThriftDocumentBuilder(FieldNameToIdMapping idMapping) {

this.idMapping = idMapping;

}

protected void prepareToBuild() {

// left empty, subclass can override this.

}

public ThriftDocument build() {

prepareToBuild();

return doc;

}

/\*\*

\* Add a long field. This is indexed as a

\* {@link com.twitter.search.common.util.analysis.LongTermAttribute}

\*/

public final ThriftDocumentBuilder withLongField(String fieldName, long value) {

ThriftFieldData fieldData = new ThriftFieldData().setLongValue(value);

ThriftField field = new ThriftField()

.setFieldConfigId(idMapping.getFieldID(fieldName)).setFieldData(fieldData);

doc.addToFields(field);

return this;

}

/\*\*

\* Add an int field. This is indexed as a

\* {@link com.twitter.search.common.util.analysis.IntTermAttribute}

\*/

public final ThriftDocumentBuilder withIntField(String fieldName, int value) {

ThriftFieldData fieldData = new ThriftFieldData().setIntValue(value);

ThriftField field = new ThriftField()

.setFieldConfigId(idMapping.getFieldID(fieldName)).setFieldData(fieldData);

doc.addToFields(field);

return this;

}

/\*\*

\* Add a field whose value is a single byte.

\*/

public final ThriftDocumentBuilder withByteField(String fieldName, byte value) {

ThriftFieldData fieldData = new ThriftFieldData().setByteValue(value);

ThriftField field = new ThriftField()

.setFieldConfigId(idMapping.getFieldID(fieldName)).setFieldData(fieldData);

doc.addToFields(field);

return this;

}

/\*\*

\* Add a field whose value is a byte array.

\*/

public final ThriftDocumentBuilder withBytesField(String fieldName, byte[] value) {

ThriftFieldData fieldData = new ThriftFieldData().setBytesValue(value);

ThriftField field = new ThriftField()

.setFieldConfigId(idMapping.getFieldID(fieldName)).setFieldData(fieldData);

doc.addToFields(field);

return this;

}

/\*\*

\* Add a field whose value is a float.

\*/

public final ThriftDocumentBuilder withFloatField(String fieldName, float value) {

ThriftFieldData fieldData = new ThriftFieldData().setFloatValue(value);

ThriftField field = new ThriftField()

.setFieldConfigId(idMapping.getFieldID(fieldName)).setFieldData(fieldData);

doc.addToFields(field);

return this;

}

/\*\*

\* Added a field whose value is a Lucene TokenStream.

\* The Lucene TokenStream is serialized using Twitter's

\* {@link com.twitter.common.text.util.TokenStreamSerializer}

\*/

public final ThriftDocumentBuilder withTokenStreamField(String fieldName,

@Nullable String tokenStreamText,

byte[] tokenStream) {

if (tokenStream == null) {

return this;

}

ThriftFieldData fieldData = new ThriftFieldData()

.setStringValue(tokenStreamText).setTokenStreamValue(tokenStream);

ThriftField field = new ThriftField()

.setFieldConfigId(idMapping.getFieldID(fieldName)).setFieldData(fieldData);

doc.addToFields(field);

return this;

}

/\*\*

\* Add a field whose value is a String.

\* @param fieldName Name of the field where the string will be added.

\* @param text This string is indexed as is (not analyzed).

\*/

public final ThriftDocumentBuilder withStringField(String fieldName, String text) {

if (text == null || text.isEmpty()) {

return this;

}

ThriftFieldData fieldData = new ThriftFieldData().setStringValue(text);

ThriftField field = new ThriftField()

.setFieldConfigId(idMapping.getFieldID(fieldName)).setFieldData(fieldData);

doc.addToFields(field);

return this;

}

/\*\*

\* Add a field whose value is a geo coordinate.

\* Earlybird will process the coordinates into geo hashes before indexing.

\*/

public final ThriftDocumentBuilder withGeoField(String fieldName,

double lat, double lon, int acc) {

if (!GeoUtil.validateGeoCoordinates(lat, lon)) {

// If the geo coordinates are invalid, don't add any field.

return this;

}

ThriftGeoCoordinate coord = new ThriftGeoCoordinate();

coord.setLat(lat);

coord.setLon(lon);

coord.setAccuracy(acc);

ThriftFieldData fieldData = new ThriftFieldData().setGeoCoordinate(coord);

ThriftField field = new ThriftField()

.setFieldConfigId(idMapping.getFieldID(fieldName)).setFieldData(fieldData);

doc.addToFields(field);

return this;

}

/\*\*

\* Added a list of tokens that are weighted. The weights are stored inside payload.

\* See {@link com.twitter.search.common.util.analysis.PayloadWeightedTokenizer} for more details.

\*/

public final ThriftDocumentBuilder withPayloadWeightTokenStreamField(String fieldName,

String tokens) {

byte[] serialized;

try {

PayloadWeightedTokenizer tokenizer = new PayloadWeightedTokenizer(tokens);

serialized = PAYLOAD\_WEIGHTED\_SERIALIZER\_PER\_THREAD.get().serialize(tokenizer);

tokenizer.close();

} catch (IOException e) {

LOG.log(Level.WARNING,

"Failed to add PayloadWeightedTokenizer field. Bad token weight list: " + tokens, e);

return this;

} catch (NumberFormatException e) {

LOG.log(Level.WARNING,

"Failed to add PayloadWeightedTokenizer field. Cannot parse token weight: " + tokens, e);

return this;

}

withTokenStreamField(fieldName, tokens, serialized);

return this;

}

/\*\*

\* Add a field whose value is a list of longs.

\* Each long is encoded into a LongTermAttribute.

\* The field will contain a LongTermTokenStream.

\*/

public final ThriftDocumentBuilder withLongIDsField(String fieldName,

List<Long> longList) throws IOException {

if (longList == null || longList.isEmpty()) {

return this;

}

LongTermsTokenStream stream = new LongTermsTokenStream(longList);

stream.reset();

byte[] serializedStream = LONG\_TERM\_SERIALIZER\_PER\_THREAD.get().serialize(stream);

ThriftFieldData fieldData = new ThriftFieldData().setTokenStreamValue(serializedStream);

ThriftField field = new ThriftField()

.setFieldConfigId(idMapping.getFieldID(fieldName)).setFieldData(fieldData);

doc.addToFields(field);

return this;

}

}