package com.twitter.search.common.schema.base;

import java.util.Collection;

import java.util.Map;

import javax.annotation.Nullable;

import com.google.common.base.Predicate;

import com.google.common.collect.ImmutableCollection;

import com.google.common.collect.ImmutableMap;

import org.apache.lucene.analysis.Analyzer;

import org.apache.lucene.facet.FacetsConfig;

import org.apache.lucene.index.FieldInfos;

import com.twitter.search.common.features.thrift.ThriftSearchFeatureSchema;

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

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

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

/\*\*

\* Search Schema.

\*/

public interface Schema {

/\*\*

\* Certain Schema implementations can evolve at run time. This call returns a snapshot of

\* of the schema which is guaranteed to not change.

\*/

ImmutableSchemaInterface getSchemaSnapshot();

/\*\*

\* Returns a string describing the current schema version.

\*/

String getVersionDescription();

/\*\*

\* Returns whether the schema version is official. Only official segments are uploaded to HDFS.

\*/

boolean isVersionOfficial();

/\*\*

\* Returns the schema's major version.

\*/

int getMajorVersionNumber();

/\*\*

\* Returns the schema's minor version.

\*/

int getMinorVersionNumber();

/\*\*

\* Returns the default analyzer. This analyzer is used when none is specified on the field info.

\*/

Analyzer getDefaultAnalyzer(ThriftAnalyzer override);

/\*\*

\* Returns whether the given field is configured in the schema.

\*/

boolean hasField(int fieldConfigId);

/\*\*

\* Returns whether the given field is configured in the schema.

\*/

boolean hasField(String fieldName);

/\*\*

\* Get the field name corresponding to the given field id.

\*/

String getFieldName(int fieldConfigId);

/\*\*

\* Return the FieldInfo of all fields.

\*/

ImmutableCollection<FieldInfo> getFieldInfos();

/\*\*

\* Get the field info for the given field id. If an override is given, attempt to merge the

\* base field info with the override config.

\*/

FieldInfo getFieldInfo(int fieldConfigId, ThriftFieldConfiguration override);

/\*\*

\* Get the field info for the given field id. No override.

\*/

@Nullable

FieldInfo getFieldInfo(int fieldConfigId);

/\*\*

\* Get the field info for the given field name. No override.

\*/

@Nullable

FieldInfo getFieldInfo(String fieldName);

/\*\*

\* Builds a lucene FieldInfos instance, usually used for indexing.

\*/

FieldInfos getLuceneFieldInfos(Predicate<String> acceptedFields);

/\*\*

\* Returns the number of facet fields in this schema.

\*/

int getNumFacetFields();

/\*\*

\* Return facet configurations.

\*/

FacetsConfig getFacetsConfig();

/\*\*

\* Get the facet field's field info by facet name.

\*/

FieldInfo getFacetFieldByFacetName(String facetName);

/\*\*

\* Get the facet field's field info by field name.

\*/

FieldInfo getFacetFieldByFieldName(String fieldName);

/\*\*

\* Get the field infos for all facet fields.

\*/

Collection<FieldInfo> getFacetFields();

/\*\*

\* Get the field infos for all facet fields backed by column stride fields.

\*/

Collection<FieldInfo> getCsfFacetFields();

/\*\*

\* Get the field weight map for text searchable fields.

\*/

Map<String, FieldWeightDefault> getFieldWeightMap();

/\*\*

\* Get scoring feature configuration by feature name.

\*/

FeatureConfiguration getFeatureConfigurationByName(String featureName);

/\*\*

\* Get scoring feature configuration by feature field id. The feature configuration is

\* guaranteed to be not null, or a NullPointerException will be thrown out.

\*/

FeatureConfiguration getFeatureConfigurationById(int featureFieldId);

/\*\*

\* Returns the ThriftCSFType for a CSF field.

\* Note: for non-CSF field, null will be returned.

\*/

@Nullable

ThriftCSFType getCSFFieldType(String fieldName);

/\*\*

\* Get the search result feature schema for all possible features in all search results.

\*

\* The returned value is not really immutable (because it's a pre-generated thrift struct).

\* We want to return it directly because we want to pre-build it once and return with the thrift

\* search results as is.

\*/

ThriftSearchFeatureSchema getSearchFeatureSchema();

/\*\*

\* Get the mapping from feature id to feature configuration.

\*/

ImmutableMap<Integer, FeatureConfiguration> getFeatureIdToFeatureConfig();

/\*\*

\* Get the mapping from feature name to feature configuration.

\*/

ImmutableMap<String, FeatureConfiguration> getFeatureNameToFeatureConfig();

/\*\*

\* Field configuration for a single field.

\*/

final class FieldInfo {

private final int fieldId;

private final String name;

private final EarlybirdFieldType luceneFieldType;

public FieldInfo(int fieldId, String name, EarlybirdFieldType luceneFieldType) {

this.fieldId = fieldId;

this.name = name;

this.luceneFieldType = luceneFieldType;

}

public int getFieldId() {

return fieldId;

}

public String getName() {

return name;

}

public EarlybirdFieldType getFieldType() {

return luceneFieldType;

}

public String getDescription() {

return String.format(

"(FieldInfo [fieldId: %d, name: %s, luceneFieldType: %s])",

fieldId, name, luceneFieldType.getFacetName()

);

}

@Override

public boolean equals(Object obj) {

if (!(obj instanceof FieldInfo)) {

return false;

}

return fieldId == ((FieldInfo) obj).fieldId;

}

@Override

public int hashCode() {

return fieldId;

}

}

/\*\*

\* Exception thrown when errors or inconsistences are detected in a search schema.

\*/

final class SchemaValidationException extends Exception {

public SchemaValidationException(String msg) {

super(msg);

}

public SchemaValidationException(String msg, Exception e) {

super(msg, e);

}

}

}