package com.twitter.search.earlybird.search;

import java.util.List;

import javax.annotation.Nullable;

import com.google.common.base.Preconditions;

import org.apache.lucene.search.Query;

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

import com.twitter.search.common.query.HitAttributeHelper;

import com.twitter.search.common.search.TerminationTracker;

import com.twitter.search.earlybird.QualityFactor;

import com.twitter.search.earlybird.thrift.ThriftSearchQuery;

import com.twitter.search.queryparser.util.IdTimeRanges;

public class SearchRequestInfo {

private final ThriftSearchQuery searchQuery;

private final Query luceneQuery;

private final boolean collectConversationId;

private final boolean collectResultLocation;

private final boolean getInReplyToStatusId;

private final boolean getReferenceAuthorId;

private final boolean getFromUserId;

private final boolean collectExclusiveConversationAuthorId;

private final int numResultsRequested;

private final int maxHitsToProcess;

private final List<String> facetFieldNames;

private long timestamp;

private final TerminationTracker terminationTracker;

protected final QualityFactor qualityFactor;

// Set if we want to collect per-field hit attributes for this request.

@Nullable

private HitAttributeHelper hitAttributeHelper;

private IdTimeRanges idTimeRanges;

private static final int DEFAULT\_MAX\_HITS = 1000;

private static final SearchCounter RESET\_MAX\_HITS\_TO\_PROCESS\_COUNTER =

SearchCounter.export("search\_request\_info\_reset\_max\_hits\_to\_process");

public SearchRequestInfo(

ThriftSearchQuery searchQuery,

Query luceneQuery,

TerminationTracker terminationTracker) {

this(searchQuery, luceneQuery, terminationTracker, null);

}

public SearchRequestInfo(

ThriftSearchQuery searchQuery,

Query luceneQuery,

TerminationTracker terminationTracker,

QualityFactor qualityFactor) {

Preconditions.checkNotNull(searchQuery.getCollectorParams());

Preconditions.checkNotNull(terminationTracker);

this.searchQuery = searchQuery;

this.luceneQuery = luceneQuery;

this.collectConversationId = searchQuery.isCollectConversationId();

if (searchQuery.isSetResultMetadataOptions()) {

this.collectResultLocation = searchQuery.getResultMetadataOptions().isGetResultLocation();

this.getInReplyToStatusId = searchQuery.getResultMetadataOptions().isGetInReplyToStatusId();

this.getReferenceAuthorId =

searchQuery.getResultMetadataOptions().isGetReferencedTweetAuthorId();

this.getFromUserId = searchQuery.getResultMetadataOptions().isGetFromUserId();

this.collectExclusiveConversationAuthorId =

searchQuery.getResultMetadataOptions().isGetExclusiveConversationAuthorId();

} else {

this.collectResultLocation = false;

this.getInReplyToStatusId = false;

this.getReferenceAuthorId = false;

this.getFromUserId = false;

this.collectExclusiveConversationAuthorId = false;

}

this.qualityFactor = qualityFactor;

this.numResultsRequested = searchQuery.getCollectorParams().getNumResultsToReturn();

this.maxHitsToProcess = calculateMaxHitsToProcess(searchQuery);

this.terminationTracker = terminationTracker;

this.facetFieldNames = searchQuery.getFacetFieldNames();

}

/\*\*

\* Gets the value to be used as max hits to process for this query. The base class gets it from

\* the searchQuery directly, and uses a default if that's not set.

\*

\* Subclasses can override this to compute a different value for max hits to process.

\*/

protected int calculateMaxHitsToProcess(ThriftSearchQuery thriftSearchQuery) {

int maxHits = thriftSearchQuery.getCollectorParams().isSetTerminationParams()

? thriftSearchQuery.getCollectorParams().getTerminationParams().getMaxHitsToProcess() : 0;

if (maxHits <= 0) {

maxHits = DEFAULT\_MAX\_HITS;

RESET\_MAX\_HITS\_TO\_PROCESS\_COUNTER.increment();

}

return maxHits;

}

public final ThriftSearchQuery getSearchQuery() {

return this.searchQuery;

}

public Query getLuceneQuery() {

return luceneQuery;

}

public final int getNumResultsRequested() {

return numResultsRequested;

}

public final int getMaxHitsToProcess() {

return maxHitsToProcess;

}

public boolean isCollectConversationId() {

return collectConversationId;

}

public boolean isCollectResultLocation() {

return collectResultLocation;

}

public boolean isGetInReplyToStatusId() {

return getInReplyToStatusId;

}

public boolean isGetReferenceAuthorId() {

return getReferenceAuthorId;

}

public boolean isCollectExclusiveConversationAuthorId() {

return collectExclusiveConversationAuthorId;

}

public final IdTimeRanges getIdTimeRanges() {

return idTimeRanges;

}

public SearchRequestInfo setIdTimeRanges(IdTimeRanges newIdTimeRanges) {

this.idTimeRanges = newIdTimeRanges;

return this;

}

public SearchRequestInfo setTimestamp(long newTimestamp) {

this.timestamp = newTimestamp;

return this;

}

public long getTimestamp() {

return timestamp;

}

public TerminationTracker getTerminationTracker() {

return this.terminationTracker;

}

@Nullable

public HitAttributeHelper getHitAttributeHelper() {

return hitAttributeHelper;

}

public void setHitAttributeHelper(@Nullable HitAttributeHelper hitAttributeHelper) {

this.hitAttributeHelper = hitAttributeHelper;

}

public List<String> getFacetFieldNames() {

return facetFieldNames;

}

public boolean isGetFromUserId() {

return getFromUserId;

}

}