package com.twitter.search.earlybird\_root.common;

import javax.annotation.Nonnull;

import com.twitter.search.common.constants.thriftjava.ThriftQuerySource;

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

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

/\*\*

\* Earlybird roots distinguish these types of requests and treat them differently.

\*/

public enum EarlybirdRequestType {

FACETS,

RECENCY,

RELEVANCE,

STRICT\_RECENCY,

TERM\_STATS,

TOP\_TWEETS;

/\*\*

\* Returns the type of the given requests.

\*/

@Nonnull

public static EarlybirdRequestType of(EarlybirdRequest request) {

if (request.isSetFacetRequest()) {

return FACETS;

} else if (request.isSetTermStatisticsRequest()) {

return TERM\_STATS;

} else if (request.isSetSearchQuery() && request.getSearchQuery().isSetRankingMode()) {

ThriftSearchRankingMode rankingMode = request.getSearchQuery().getRankingMode();

switch (rankingMode) {

case RECENCY:

if (shouldUseStrictRecency(request)) {

return STRICT\_RECENCY;

} else {

return RECENCY;

}

case RELEVANCE:

return RELEVANCE;

case TOPTWEETS:

return TOP\_TWEETS;

default:

throw new IllegalArgumentException();

}

} else {

throw new UnsupportedOperationException();

}

}

private static boolean shouldUseStrictRecency(EarlybirdRequest request) {

// For now, we decide to do strict merging solely based on the QuerySource, and only for GNIP.

return request.isSetQuerySource() && request.getQuerySource() == ThriftQuerySource.GNIP;

}

private final String normalizedName;

EarlybirdRequestType() {

this.normalizedName = name().toLowerCase();

}

/\*\*

\* Returns the "normalized" name of this request type, that can be used for stat and decider

\* names.

\*/

public String getNormalizedName() {

return normalizedName;

}

}