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

import com.google.common.base.Optional;

import com.google.common.base.Preconditions;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import com.twitter.search.common.caching.CacheUtil;

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

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

import com.twitter.search.earlybird\_root.common.EarlybirdRequestContext;

import com.twitter.search.queryparser.query.Query;

import com.twitter.search.queryparser.query.QueryParserException;

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

public class RecencyAndRelevanceCachePostProcessor extends EarlybirdCachePostProcessor {

private static final Logger LOG =

LoggerFactory.getLogger(RecencyAndRelevanceCachePostProcessor.class);

protected Optional<EarlybirdResponse> postProcessCacheResponse(

EarlybirdRequest earlybirdRequest,

EarlybirdResponse earlybirdResponse, long sinceID, long maxID) {

return CacheUtil.postProcessCacheResult(

earlybirdRequest, earlybirdResponse, sinceID, maxID);

}

@Override

public final Optional<EarlybirdResponse> processCacheResponse(

EarlybirdRequestContext requestContext,

EarlybirdResponse cacheResponse) {

EarlybirdRequest originalRequest = requestContext.getRequest();

Preconditions.checkArgument(originalRequest.isSetSearchQuery());

IdTimeRanges ranges;

Query query = requestContext.getParsedQuery();

if (query != null) {

try {

ranges = IdTimeRanges.fromQuery(query);

} catch (QueryParserException e) {

LOG.error(

"Exception when parsing since and max IDs. Request: {} Response: {}",

originalRequest,

cacheResponse,

e);

return Optional.absent();

}

} else {

ranges = null;

}

Optional<Long> sinceID;

Optional<Long> maxID;

if (ranges != null) {

sinceID = ranges.getSinceIDExclusive();

maxID = ranges.getMaxIDInclusive();

} else {

sinceID = Optional.absent();

maxID = Optional.absent();

}

return postProcessCacheResponse(

originalRequest, cacheResponse, sinceID.or(0L), maxID.or(Long.MAX\_VALUE));

}

}