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

import javax.inject.Inject;

import javax.inject.Named;

import com.twitter.common.util.Clock;

import com.twitter.finagle.Service;

import com.twitter.search.common.decider.SearchDecider;

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

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

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

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

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

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

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

import com.twitter.search.earlybird\_root.filters.EarlybirdTimeRangeFilter;

public class RecencyRequestRouter extends AbstractRecencyAndRelevanceRequestRouter {

private static final SearchCounter SKIPPED\_ARCHIVE\_DUE\_TO\_REALTIME\_EARLY\_TERMINATION\_COUNTER =

SearchCounter.export("recency\_skipped\_archive\_due\_to\_realtime\_early\_termination");

private static final SearchCounter SKIPPED\_ARCHIVE\_DUE\_TO\_REALTIME\_ENOUGH\_RESULTS\_COUNTER =

SearchCounter.export("recency\_skipped\_archive\_due\_to\_realtime\_enough\_results");

@Inject

public RecencyRequestRouter(

@Named(InjectionNames.REALTIME)

Service<EarlybirdRequestContext, EarlybirdResponse> realtime,

@Named(InjectionNames.PROTECTED)

Service<EarlybirdRequestContext, EarlybirdResponse> protectedRealtime,

@Named(InjectionNames.FULL\_ARCHIVE)

Service<EarlybirdRequestContext, EarlybirdResponse> fullArchive,

@Named(RecencyRequestRouterModule.REALTIME\_TIME\_RANGE\_FILTER)

EarlybirdTimeRangeFilter realtimeTimeRangeFilter,

@Named(RecencyRequestRouterModule.PROTECTED\_TIME\_RANGE\_FILTER)

EarlybirdTimeRangeFilter protectedTimeRangeFilter,

@Named(RecencyRequestRouterModule.FULL\_ARCHIVE\_TIME\_RANGE\_FILTER)

EarlybirdTimeRangeFilter fullArchiveTimeRangeFilter,

Clock clock,

SearchDecider decider,

EarlybirdFeatureSchemaMerger featureSchemaMerger) {

super(realtime,

protectedRealtime,

fullArchive,

realtimeTimeRangeFilter,

protectedTimeRangeFilter,

fullArchiveTimeRangeFilter,

ThriftSearchRankingMode.RECENCY,

clock,

decider,

featureSchemaMerger);

}

@Override

protected boolean shouldSendRequestToFullArchiveCluster(

EarlybirdRequest request, EarlybirdResponse realtimeResponse) {

boolean isEarlyTerminated = realtimeResponse.isSetEarlyTerminationInfo()

&& realtimeResponse.getEarlyTerminationInfo().isEarlyTerminated();

if (isEarlyTerminated) {

SKIPPED\_ARCHIVE\_DUE\_TO\_REALTIME\_EARLY\_TERMINATION\_COUNTER.increment();

return false;

}

// Check if we have the minimum number of results to fulfill the original request.

int numResultsRequested = request.getSearchQuery().getNumResults();

int actualNumResults = realtimeResponse.getSearchResults().getResultsSize();

if (actualNumResults >= numResultsRequested) {

SKIPPED\_ARCHIVE\_DUE\_TO\_REALTIME\_ENOUGH\_RESULTS\_COUNTER.increment();

return false;

}

return true;

}

}