package com.twitter.search.earlybird\_root;

import scala.runtime.AbstractFunction0;

import com.twitter.common.util.Clock;

import com.twitter.finagle.thrift.ClientId;

import com.twitter.search.common.caching.thriftjava.CachingParams;

import com.twitter.search.common.query.thriftjava.CollectorParams;

import com.twitter.search.common.ranking.thriftjava.ThriftRankingParams;

import com.twitter.search.common.ranking.thriftjava.ThriftScoringFunctionType;

import com.twitter.search.common.root.SearchRootWarmup;

import com.twitter.search.common.root.WarmupConfig;

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

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

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

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

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

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

import com.twitter.util.Future;

/\*\*

\* Warm-up logic for Earlybird Roots.

\* Sends 60 rounds of requests with a 1 second timeout between each round.

\* The actual number of requests sent by each round can be configured.

\*/

public class EarlybirdWarmup extends

SearchRootWarmup<EarlybirdService.ServiceIface, EarlybirdRequest, EarlybirdResponse> {

private static final int WARMUP\_NUM\_RESULTS = 20;

private static final String CLIENT\_ID = "earlybird\_root\_warmup";

public EarlybirdWarmup(Clock clock, WarmupConfig config) {

super(clock, config);

}

@Override

protected EarlybirdRequest createRequest(int requestId) {

String query = "(\* " + "warmup" + requestId + ")";

return new EarlybirdRequest()

.setSearchQuery(

new ThriftSearchQuery()

.setNumResults(WARMUP\_NUM\_RESULTS)

.setCollectorParams(

new CollectorParams().setNumResultsToReturn(WARMUP\_NUM\_RESULTS))

.setRankingMode(ThriftSearchRankingMode.RELEVANCE)

.setRelevanceOptions(new ThriftSearchRelevanceOptions()

.setRankingParams(new ThriftRankingParams()

.setType(ThriftScoringFunctionType.LINEAR)))

.setSerializedQuery(query))

.setCachingParams(new CachingParams().setCache(false))

.setClientId(CLIENT\_ID);

}

@Override

protected Future<EarlybirdResponse> callService(

final EarlybirdService.ServiceIface service,

final EarlybirdRequest request) {

return ClientId.apply(CLIENT\_ID).asCurrent(

new AbstractFunction0<Future<EarlybirdResponse>>() {

@Override

public Future<EarlybirdResponse> apply() {

return service.search(request);

}

});

}

}