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

import java.util.Date;

import java.util.concurrent.TimeUnit;

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

import com.twitter.search.common.partitioning.snowflakeparser.SnowflakeIdParser;

import com.twitter.search.common.util.date.DateUtil;

import com.twitter.search.earlybird.config.ServingRange;

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

public class FullArchiveServingRangeProvider implements ServingRangeProvider {

public static final Date FULL\_ARCHIVE\_START\_DATE = DateUtil.toDate(2006, 3, 21);

private static final int DEFAULT\_SERVING\_RANGE\_BOUNDARY\_HOURS\_AGO = 48;

private final SearchDecider decider;

private final String deciderKey;

public FullArchiveServingRangeProvider(

SearchDecider decider, String deciderKey) {

this.decider = decider;

this.deciderKey = deciderKey;

}

@Override

public ServingRange getServingRange(

final EarlybirdRequestContext requestContext, boolean useBoundaryOverride) {

return new ServingRange() {

@Override

public long getServingRangeSinceId() {

// we use 1 instead of 0, because the since\_id operator is inclusive in earlybirds.

return 1L;

}

@Override

public long getServingRangeMaxId() {

long servingRangeEndMillis = TimeUnit.HOURS.toMillis(

(decider.featureExists(deciderKey))

? decider.getAvailability(deciderKey)

: DEFAULT\_SERVING\_RANGE\_BOUNDARY\_HOURS\_AGO);

long boundaryTime = requestContext.getCreatedTimeMillis() - servingRangeEndMillis;

return SnowflakeIdParser.generateValidStatusId(boundaryTime, 0);

}

@Override

public long getServingRangeSinceTimeSecondsFromEpoch() {

return FULL\_ARCHIVE\_START\_DATE.getTime() / 1000;

}

@Override

public long getServingRangeUntilTimeSecondsFromEpoch() {

long servingRangeEndMillis = TimeUnit.HOURS.toMillis(

(decider.featureExists(deciderKey))

? decider.getAvailability(deciderKey)

: DEFAULT\_SERVING\_RANGE\_BOUNDARY\_HOURS\_AGO);

long boundaryTime = requestContext.getCreatedTimeMillis() - servingRangeEndMillis;

return boundaryTime / 1000;

}

};

}

}