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

import java.util.concurrent.TimeUnit;

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

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

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

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

public class RealtimeServingRangeProvider implements ServingRangeProvider {

private static final int DEFAULT\_SERVING\_RANGE\_BOUNDARY\_HOURS\_AGO = 240;

private final SearchDecider decider;

private final String deciderKey;

public RealtimeServingRangeProvider(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() {

long servingRangeStartMillis = TimeUnit.HOURS.toMillis(

(decider.featureExists(deciderKey))

? decider.getAvailability(deciderKey)

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

long boundaryTime = requestContext.getCreatedTimeMillis() - servingRangeStartMillis;

return SnowflakeIdParser.generateValidStatusId(boundaryTime, 0);

}

@Override

public long getServingRangeMaxId() {

return SnowflakeIdParser.generateValidStatusId(

requestContext.getCreatedTimeMillis(), 0);

}

@Override

public long getServingRangeSinceTimeSecondsFromEpoch() {

long servingRangeStartMillis = TimeUnit.HOURS.toMillis(

(decider.featureExists(deciderKey))

? decider.getAvailability(deciderKey)

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

long boundaryTime = requestContext.getCreatedTimeMillis() - servingRangeStartMillis;

return boundaryTime / 1000;

}

@Override

public long getServingRangeUntilTimeSecondsFromEpoch() {

return requestContext.getCreatedTimeMillis() / 1000;

}

};

}

}