package com.twitter.search.earlybird.segment;

import java.util.Optional;

import com.twitter.search.common.indexing.thriftjava.ThriftVersionedEvents;

import com.twitter.search.common.util.io.EmptyRecordReader;

import com.twitter.search.common.util.io.recordreader.RecordReader;

import com.twitter.search.earlybird.document.TweetDocument;

import com.twitter.search.earlybird.partition.SegmentInfo;

/\*\*

\* A SegmentDataReaderSet that returns no data. Uses a DocumentReader that is

\* always caught up, but never gets exhausted.

\* Can be used for bringing up an earlybird against a static set of segments,

\* and will not incorporate any new updates.

\*/

public class EmptySegmentDataReaderSet implements SegmentDataReaderSet {

public static final EmptySegmentDataReaderSet INSTANCE = new EmptySegmentDataReaderSet();

@Override

public void attachDocumentReaders(SegmentInfo segmentInfo) {

}

@Override

public void attachUpdateReaders(SegmentInfo segmentInfo) {

}

@Override

public void completeSegmentDocs(SegmentInfo segmentInfo) {

}

@Override

public void stopSegmentUpdates(SegmentInfo segmentInfo) {

}

@Override

public void stopAll() {

}

@Override

public boolean allCaughtUp() {

// ALWAYS CAUGHT UP

return true;

}

@Override

public RecordReader<TweetDocument> newDocumentReader(SegmentInfo segmentInfo)

throws Exception {

return null;

}

@Override

public RecordReader<TweetDocument> getDocumentReader() {

return new EmptyRecordReader<>();

}

@Override

public RecordReader<ThriftVersionedEvents> getUpdateEventsReader() {

return null;

}

@Override

public RecordReader<ThriftVersionedEvents> getUpdateEventsReaderForSegment(

SegmentInfo segmentInfo) {

return null;

}

@Override

public Optional<Long> getUpdateEventsStreamOffsetForSegment(SegmentInfo segmentInfo) {

return Optional.of(0L);

}

}