package com.twitter.search.earlybird.search.queries;

import java.io.IOException;

import org.apache.lucene.index.LeafReaderContext;

import org.apache.lucene.index.PostingsEnum;

import org.apache.lucene.index.TermsEnum;

import org.apache.lucene.search.ConstantScoreScorer;

import org.apache.lucene.search.ConstantScoreWeight;

import org.apache.lucene.search.IndexSearcher;

import org.apache.lucene.search.Query;

import org.apache.lucene.search.Scorer;

import org.apache.lucene.search.ScoreMode;

import org.apache.lucene.search.Weight;

/\*\*

\* A version of a term query that we can use when we already know the term id (in case where we

\* previously looked it up), and have a TermsEnum to get the actual postings.

\*

\* This is can be used for constant score queries, where only iterating on the postings is required.

\*/

class SimpleTermQuery extends Query {

private final TermsEnum termsEnum;

private final long termId;

public SimpleTermQuery(TermsEnum termsEnum, long termId) {

this.termsEnum = termsEnum;

this.termId = termId;

}

@Override

public Weight createWeight(IndexSearcher searcher, ScoreMode scoreMode, float boost)

throws IOException {

return new SimpleTermQueryWeight(scoreMode);

}

@Override

public int hashCode() {

return (termsEnum == null ? 0 : termsEnum.hashCode()) \* 13 + (int) termId;

}

@Override

public boolean equals(Object obj) {

if (!(obj instanceof SimpleTermQuery)) {

return false;

}

SimpleTermQuery query = SimpleTermQuery.class.cast(obj);

return (termsEnum == null ? query.termsEnum == null : termsEnum.equals(query.termsEnum))

&& (termId == query.termId);

}

@Override

public String toString(String field) {

return "SimpleTermQuery(" + field + ":" + termId + ")";

}

private class SimpleTermQueryWeight extends ConstantScoreWeight {

private final ScoreMode scoreMode;

public SimpleTermQueryWeight(ScoreMode scoreMode) {

super(SimpleTermQuery.this, 1.0f);

this.scoreMode = scoreMode;

}

@Override

public String toString() {

return "weight(" + SimpleTermQuery.this + ")";

}

@Override

public Scorer scorer(LeafReaderContext context) throws IOException {

termsEnum.seekExact(termId);

PostingsEnum docs = termsEnum.postings(

null, scoreMode.needsScores() ? PostingsEnum.FREQS : PostingsEnum.NONE);

assert docs != null;

return new ConstantScoreScorer(this, 0, scoreMode, docs);

}

@Override

public boolean isCacheable(LeafReaderContext ctx) {

return true;

}

}

}