package com.twitter.search.common.query;

import java.io.IOException;

import java.util.Set;

import com.google.common.base.Preconditions;

import org.apache.lucene.index.LeafReaderContext;

import org.apache.lucene.index.Term;

import org.apache.lucene.search.Explanation;

import org.apache.lucene.search.Scorer;

import org.apache.lucene.search.Weight;

/\*\*

\* Weight implementation that adds attribute collection support for an underlying query.

\* Meant to be used in conjunction with {@link IdentifiableQuery}.

\*/

public class IdentifiableQueryWeight extends Weight {

private final Weight inner;

private final FieldRankHitInfo queryId;

private final HitAttributeCollector attrCollector;

/\*\* Creates a new IdentifiableQueryWeight instance. \*/

public IdentifiableQueryWeight(IdentifiableQuery query, Weight inner, FieldRankHitInfo queryId,

HitAttributeCollector attrCollector) {

super(query);

this.inner = inner;

this.queryId = queryId;

this.attrCollector = Preconditions.checkNotNull(attrCollector);

}

@Override

public Explanation explain(LeafReaderContext context, int doc)

throws IOException {

return inner.explain(context, doc);

}

@Override

public Scorer scorer(LeafReaderContext context) throws IOException {

attrCollector.clearHitAttributions(context, queryId);

Scorer innerScorer = inner.scorer(context);

if (innerScorer != null) {

return new IdentifiableQueryScorer(this, innerScorer, queryId, attrCollector);

} else {

return null;

}

}

@Override

public void extractTerms(Set<Term> terms) {

inner.extractTerms(terms);

}

@Override

public boolean isCacheable(LeafReaderContext ctx) {

return inner.isCacheable(ctx);

}

}