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

import javax.inject.Inject;

import com.google.common.annotations.VisibleForTesting;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import com.twitter.finagle.Service;

import com.twitter.finagle.SimpleFilter;

import com.twitter.search.common.metrics.SearchCounter;

import com.twitter.search.earlybird.thrift.EarlybirdResponse;

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

import com.twitter.search.queryparser.query.Query;

import com.twitter.search.queryparser.query.QueryParserException;

import com.twitter.search.queryparser.visitors.DropAllProtectedOperatorVisitor;

import com.twitter.util.Future;

public class DropAllProtectedOperatorFilter

extends SimpleFilter<EarlybirdRequestContext, EarlybirdResponse> {

private static final Logger LOG =

LoggerFactory.getLogger(DropAllProtectedOperatorFilter.class);

private static final SearchCounter QUERY\_PARSER\_FAILURE\_COUNTER =

SearchCounter.export("protected\_operator\_filter\_query\_parser\_failure\_count");

@VisibleForTesting

static final SearchCounter TOTAL\_REQUESTS\_COUNTER =

SearchCounter.export("drop\_all\_protected\_operator\_filter\_total");

@VisibleForTesting

static final SearchCounter OPERATOR\_DROPPED\_REQUESTS\_COUNTER =

SearchCounter.export("drop\_all\_protected\_operator\_filter\_operator\_dropped");

private final DropAllProtectedOperatorVisitor dropProtectedOperatorVisitor;

@Inject

public DropAllProtectedOperatorFilter(

DropAllProtectedOperatorVisitor dropProtectedOperatorVisitor

) {

this.dropProtectedOperatorVisitor = dropProtectedOperatorVisitor;

}

@Override

public Future<EarlybirdResponse> apply(

EarlybirdRequestContext requestContext,

Service<EarlybirdRequestContext, EarlybirdResponse> service) {

TOTAL\_REQUESTS\_COUNTER.increment();

Query query = requestContext.getParsedQuery();

if (query == null) {

return service.apply(requestContext);

}

Query processedQuery = query;

try {

processedQuery = query.accept(dropProtectedOperatorVisitor);

} catch (QueryParserException e) {

// this should not happen since we already have a parsed query

QUERY\_PARSER\_FAILURE\_COUNTER.increment();

LOG.warn(

"Failed to drop protected operator for serialized query: " + query.serialize(), e);

}

if (processedQuery == query) {

return service.apply(requestContext);

} else {

OPERATOR\_DROPPED\_REQUESTS\_COUNTER.increment();

EarlybirdRequestContext clonedRequestContext =

EarlybirdRequestContext.copyRequestContext(requestContext, processedQuery);

return service.apply(clonedRequestContext);

}

}

}