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

import java.util.concurrent.TimeUnit;

import javax.inject.Inject;

import com.twitter.common\_internal.text.version.PenguinVersion;

import com.twitter.common\_internal.text.version.PenguinVersionConfig;

import com.twitter.finagle.Service;

import com.twitter.finagle.SimpleFilter;

import com.twitter.finagle.tracing.Trace;

import com.twitter.finagle.tracing.Tracing;

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

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

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

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

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

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

import com.twitter.search.queryparser.parser.SerializedQueryParser;

import com.twitter.search.queryparser.parser.SerializedQueryParser.TokenizationOption;

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

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

import com.twitter.util.Duration;

import com.twitter.util.Future;

public class QueryTokenizerFilter extends SimpleFilter<EarlybirdRequestContext, EarlybirdResponse> {

private static final String PREFIX = "query\_tokenizer\_";

private static final SearchRateCounter SUCCESS\_COUNTER =

SearchRateCounter.export(PREFIX + "success");

private static final SearchRateCounter FAILURE\_COUNTER =

SearchRateCounter.export(PREFIX + "error");

private static final SearchRateCounter SKIPPED\_COUNTER =

SearchRateCounter.export(PREFIX + "skipped");

private static final SearchTimerStats QUERY\_TOKENIZER\_TIME =

SearchTimerStats.export(PREFIX + "time", TimeUnit.MILLISECONDS, false);

private final TokenizationOption tokenizationOption;

@Inject

public QueryTokenizerFilter(PenguinVersionConfig penguinversions) {

PenguinVersion[] supportedVersions = penguinversions

.getSupportedVersions().toArray(new PenguinVersion[0]);

tokenizationOption = new TokenizationOption(true, supportedVersions);

}

@Override

public Future<EarlybirdResponse> apply(

EarlybirdRequestContext requestContext,

Service<EarlybirdRequestContext, EarlybirdResponse> service) {

if (!requestContext.getRequest().isRetokenizeSerializedQuery()

|| !requestContext.getRequest().isSetSearchQuery()

|| !requestContext.getRequest().getSearchQuery().isSetSerializedQuery()) {

SKIPPED\_COUNTER.increment();

return service.apply(requestContext);

}

SearchTimer timer = QUERY\_TOKENIZER\_TIME.startNewTimer();

try {

String serializedQuery = requestContext.getRequest().getSearchQuery().getSerializedQuery();

Query parsedQuery = reparseQuery(serializedQuery);

SUCCESS\_COUNTER.increment();

return service.apply(EarlybirdRequestContext.copyRequestContext(requestContext, parsedQuery));

} catch (QueryParserException e) {

FAILURE\_COUNTER.increment();

return QueryParsingUtils.newClientErrorResponse(requestContext.getRequest(), e);

} finally {

long elapsed = timer.stop();

QUERY\_TOKENIZER\_TIME.timerIncrement(elapsed);

Tracing trace = Trace.apply();

if (trace.isActivelyTracing()) {

trace.record(PREFIX + "time", Duration.fromMilliseconds(elapsed));

}

}

}

public Query reparseQuery(String serializedQuery) throws QueryParserException {

SerializedQueryParser parser = new SerializedQueryParser(tokenizationOption);

return parser.parse(serializedQuery);

}

/\*\*

\* Initializing the query parser can take many seconds. We initialize it at warmup so that

\* requests don't time out after we join the serverset. SEARCH-28801

\*/

public void performExpensiveInitialization() throws QueryParserException {

SerializedQueryParser queryParser = new SerializedQueryParser(tokenizationOption);

// The Korean query parser takes a few seconds on it's own to initialize.

String koreanQuery = "스포츠";

queryParser.parse(koreanQuery);

}

}