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

import java.util.Map;

import java.util.concurrent.ConcurrentHashMap;

import javax.inject.Inject;

import javax.inject.Singleton;

import com.google.common.annotations.VisibleForTesting;

import com.google.common.base.Preconditions;

import com.twitter.common.text.language.LocaleUtil;

import com.twitter.finagle.Service;

import com.twitter.finagle.SimpleFilter;

import com.twitter.search.common.constants.thriftjava.ThriftLanguage;

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

import com.twitter.search.common.util.lang.ThriftLanguageUtil;

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

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

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

import com.twitter.util.Future;

/\*\*

\* Export stats for query languages.

\*/

@Singleton

public class QueryLangStatFilter

extends SimpleFilter<EarlybirdRequestContext, EarlybirdResponse> {

public static class Config {

// We put a limit here in case an error in the client are sending us random lang codes.

private int maxNumberOfLangs;

public Config(int maxNumberOfLangs) {

this.maxNumberOfLangs = maxNumberOfLangs;

}

public int getMaxNumberOfLangs() {

return maxNumberOfLangs;

}

}

@VisibleForTesting

protected static final String LANG\_STATS\_PREFIX = "num\_queries\_in\_lang\_";

private final Config config;

private final SearchCounter allCountsForLangsOverMaxNumLang =

SearchCounter.export(LANG\_STATS\_PREFIX + "overflow");

private final ConcurrentHashMap<String, SearchCounter> langCounters =

new ConcurrentHashMap<>();

@Inject

public QueryLangStatFilter(Config config) {

this.config = config;

}

private SearchCounter getCounter(String lang) {

Preconditions.checkNotNull(lang);

SearchCounter counter = langCounters.get(lang);

if (counter == null) {

if (langCounters.size() >= config.getMaxNumberOfLangs()) {

return allCountsForLangsOverMaxNumLang;

}

synchronized (langCounters) { // This double-checked locking is safe,

// since we're using a ConcurrentHashMap

counter = langCounters.get(lang);

if (counter == null) {

counter = SearchCounter.export(LANG\_STATS\_PREFIX + lang);

langCounters.put(lang, counter);

}

}

}

return counter;

}

@Override

public Future<EarlybirdResponse> apply(

EarlybirdRequestContext requestContext,

Service<EarlybirdRequestContext, EarlybirdResponse> service) {

String lang = null;

ThriftSearchQuery searchQuery = requestContext.getRequest().getSearchQuery();

lang = searchQuery.getQueryLang();

if (lang == null) {

// fallback to ui lang

lang = searchQuery.getUiLang();

}

if (lang == null && searchQuery.isSetUserLangs()) {

// fallback to the user lang with the highest confidence

double maxConfidence = Double.MIN\_VALUE;

for (Map.Entry<ThriftLanguage, Double> entry : searchQuery.getUserLangs().entrySet()) {

if (entry.getValue() > maxConfidence) {

lang = ThriftLanguageUtil.getLanguageCodeOf(entry.getKey());

maxConfidence = entry.getValue();

}

}

}

if (lang == null) {

lang = LocaleUtil.UNDETERMINED\_LANGUAGE;

}

getCounter(lang).increment();

return service.apply(requestContext);

}

}