package com.twitter.search.earlybird.search.facets;

import java.util.LinkedHashMap;

import java.util.List;

import java.util.Map;

import com.google.common.annotations.VisibleForTesting;

import com.google.common.collect.ImmutableList;

import com.google.common.collect.ImmutableSet;

import org.apache.lucene.util.BytesRef;

import com.twitter.search.common.schema.earlybird.EarlybirdFieldConstants.EarlybirdFieldConstant;

import com.twitter.search.core.earlybird.facets.FacetLabelProvider;

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

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

import com.twitter.service.spiderduck.gen.MediaTypes;

/\*\*

\* A collector for collecting expanded urls from facets. Note that the only thing connecting this

\* collector with expanded URLs is the fact that we only store the expanded url in the facet fields.

\*/

public class ExpandedUrlCollector extends AbstractFacetTermCollector {

private static final ImmutableSet<String> FACET\_CONTAINS\_URL = ImmutableSet.of(

EarlybirdFieldConstant.VIDEOS\_FACET,

EarlybirdFieldConstant.IMAGES\_FACET,

EarlybirdFieldConstant.NEWS\_FACET,

EarlybirdFieldConstant.LINKS\_FACET,

EarlybirdFieldConstant.TWIMG\_FACET);

private final Map<String, ThriftSearchResultUrl> dedupedUrls = new LinkedHashMap<>();

@Override

protected String getTermFromProvider(

String facetName,

long termID,

FacetLabelProvider provider) {

String url = null;

if (EarlybirdFieldConstant.TWIMG\_FACET.equals(facetName)) {

// Special case extraction of media url for twimg.

FacetLabelProvider.FacetLabelAccessor photoAccessor = provider.getLabelAccessor();

BytesRef termPayload = photoAccessor.getTermPayload(termID);

if (termPayload != null) {

url = termPayload.utf8ToString();

}

} else {

url = provider.getLabelAccessor().getTermText(termID);

}

return url;

}

@Override

public boolean collect(int docID, long termID, int fieldID) {

String url = getTermFromFacet(termID, fieldID, FACET\_CONTAINS\_URL);

if (url == null || url.isEmpty()) {

return false;

}

ThriftSearchResultUrl resultUrl = new ThriftSearchResultUrl();

resultUrl.setOriginalUrl(url);

MediaTypes mediaType = getMediaType(findFacetName(fieldID));

resultUrl.setMediaType(mediaType);

// Media links will show up twice:

// - once in image/native\_image/video/news facets

// - another time in the links facet

//

// For those urls, we only want to return the media version. If it is non-media version, only

// write to map if doesn't exist already, if media version, overwrite any previous entries.

if (mediaType == MediaTypes.UNKNOWN) {

if (!dedupedUrls.containsKey(url)) {

dedupedUrls.put(url, resultUrl);

}

} else {

dedupedUrls.put(url, resultUrl);

}

return true;

}

@Override

public void fillResultAndClear(ThriftSearchResult result) {

result.getMetadata().setTweetUrls(getExpandedUrls());

dedupedUrls.clear();

}

@VisibleForTesting

List<ThriftSearchResultUrl> getExpandedUrls() {

return ImmutableList.copyOf(dedupedUrls.values());

}

/\*\*

\* Gets the Spiderduck media type for a given facet name.

\*

\* @param facetName A given facet name.

\* @return {@code MediaTypes} enum corresponding to the facet name.

\*/

private static MediaTypes getMediaType(String facetName) {

if (facetName == null) {

return MediaTypes.UNKNOWN;

}

switch (facetName) {

case EarlybirdFieldConstant.TWIMG\_FACET:

return MediaTypes.NATIVE\_IMAGE;

case EarlybirdFieldConstant.IMAGES\_FACET:

return MediaTypes.IMAGE;

case EarlybirdFieldConstant.VIDEOS\_FACET:

return MediaTypes.VIDEO;

case EarlybirdFieldConstant.NEWS\_FACET:

return MediaTypes.NEWS;

default:

return MediaTypes.UNKNOWN;

}

}

}