package com.twitter.search.common.relevance.entities;

import java.util.Locale;

import com.google.common.base.Preconditions;

import org.apache.commons.lang.StringUtils;

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

import com.twitter.search.common.indexing.thriftjava.PotentialLocation;

import com.twitter.search.common.util.text.LanguageIdentifierHelper;

import com.twitter.search.common.util.text.NormalizerHelper;

import com.twitter.search.common.util.text.TokenizerHelper;

/\*\*

\* An immutable tuple to wrap a country code, region and locality. Based on the PotentialLocation

\* struct in status.thrift.

\*/

public class PotentialLocationObject {

private final String countryCode;

private final String region;

private final String locality;

/\*\*

\* Creates a new PotentialLocationObject instance.

\*

\* @param countryCode The country code.

\* @param region The region.

\* @param locality The locality.

\*/

public PotentialLocationObject(String countryCode, String region, String locality) {

this.countryCode = countryCode;

this.region = region;

this.locality = locality;

}

public String getCountryCode() {

return countryCode;

}

public String getRegion() {

return region;

}

public String getLocality() {

return locality;

}

/\*\*

\* Converts this PotentialLocationObject instance to a PotentialLocation thrift struct.

\*

\* @param penguinVersion The penguin version to use for normalization and tokenization.

\*/

public PotentialLocation toThriftPotentialLocation(PenguinVersion penguinVersion) {

Preconditions.checkNotNull(penguinVersion);

String normalizedCountryCode = null;

if (countryCode != null) {

Locale countryCodeLocale = LanguageIdentifierHelper.identifyLanguage(countryCode);

normalizedCountryCode =

NormalizerHelper.normalize(countryCode, countryCodeLocale, penguinVersion);

}

String tokenizedRegion = null;

if (region != null) {

Locale regionLocale = LanguageIdentifierHelper.identifyLanguage(region);

String normalizedRegion = NormalizerHelper.normalize(region, regionLocale, penguinVersion);

tokenizedRegion = StringUtils.join(

TokenizerHelper.tokenizeQuery(normalizedRegion, regionLocale, penguinVersion), " ");

}

String tokenizedLocality = null;

if (locality != null) {

Locale localityLocale = LanguageIdentifierHelper.identifyLanguage(locality);

String normalizedLocality =

NormalizerHelper.normalize(locality, localityLocale, penguinVersion);

tokenizedLocality =

StringUtils.join(TokenizerHelper.tokenizeQuery(

normalizedLocality, localityLocale, penguinVersion), " ");

}

return new PotentialLocation()

.setCountryCode(normalizedCountryCode)

.setRegion(tokenizedRegion)

.setLocality(tokenizedLocality);

}

@Override

public int hashCode() {

return ((countryCode == null) ? 0 : countryCode.hashCode())

+ 13 \* ((region == null) ? 0 : region.hashCode())

+ 19 \* ((locality == null) ? 0 : locality.hashCode());

}

@Override

public boolean equals(Object obj) {

if (!(obj instanceof PotentialLocationObject)) {

return false;

}

PotentialLocationObject entry = (PotentialLocationObject) obj;

return (countryCode == null

? entry.countryCode == null

: countryCode.equals(entry.countryCode))

&& (region == null

? entry.region == null

: region.equals(entry.region))

&& (locality == null

? entry.locality == null

: locality.equals(entry.locality));

}

@Override

public String toString() {

return new StringBuilder("PotentialLocationObject {")

.append("countryCode=").append(countryCode)

.append(", region=").append(region)

.append(", locality=").append(locality)

.append("}")

.toString();

}

}