package com.twitter.search.common.util.lang;

import java.lang.reflect.Field;

import java.util.Locale;

import java.util.Map;

import javax.annotation.Nullable;

import com.google.common.base.Preconditions;

import com.google.common.collect.ImmutableMap;

import com.google.common.collect.Maps;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

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

/\*\*

\* This class can be used to convert ThriftLanguage to Locale object and vise versa.

\*/

public final class ThriftLanguageUtil {

private static final Logger LOG = LoggerFactory.getLogger(ThriftLanguageUtil.class.getName());

// stores ThriftLanguage.id -> Locale mapping

private static final Locale[] LOCALES;

// stores Locale -> ThriftLanguage mapping

private static final Map<Locale, ThriftLanguage> THRIFT\_LANGUAGES;

static {

LOCALES = new Locale[ThriftLanguage.values().length];

Map<Locale, ThriftLanguage> thriftLanguageMap = Maps.newHashMap();

// get all languages defined in ThriftLanguage

Field[] fields = ThriftLanguage.class.getDeclaredFields();

for (Field field : fields) {

if (!field.isEnumConstant()) {

continue;

}

try {

ThriftLanguage thriftLang = (ThriftLanguage) field.get(null);

String thriftLanguageName = field.getName();

// get corresponding Locale declared in LocaleUtil

try {

Field localeUtilField = LocaleUtil.class.getDeclaredField(thriftLanguageName);

Locale localeLang = (Locale) localeUtilField.get(null);

LOCALES[thriftLang.getValue()] = localeLang;

thriftLanguageMap.put(localeLang, thriftLang);

} catch (NoSuchFieldException e) {

LOG.warn("{} is defined in ThriftLanguage, but not in LocaleUtil.", thriftLanguageName);

}

} catch (IllegalAccessException e) {

// shouldn't happen.

LOG.warn("Could not get a declared field.", e);

}

}

// Let's make sure that all Locales defined in LocaleUtil are also defined in ThriftLanguage

for (Locale lang : LocaleUtil.getDefinedLanguages()) {

if (!thriftLanguageMap.containsKey(lang)) {

LOG.warn("{} is defined in LocaleUtil but not in ThriftLanguage.", lang.getLanguage());

}

}

THRIFT\_LANGUAGES = ImmutableMap.copyOf(thriftLanguageMap);

}

private ThriftLanguageUtil() {

}

/\*\*

\* Returns a Locale object which corresponds to a given ThriftLanguage object.

\* @param language ThriftLanguage object

\* @return a corresponding Locale object

\*/

public static Locale getLocaleOf(ThriftLanguage language) {

// Note that ThriftLanguage.findByValue() can return null (thrift generated code).

// So ThriftLanguageUtil.getLocaleOf needs to handle null correctly.

if (language == null) {

return LocaleUtil.UNKNOWN;

}

Preconditions.checkArgument(language.getValue() < LOCALES.length);

return LOCALES[language.getValue()];

}

/\*\*

\* Returns a ThriftLanguage object which corresponds to a given Locale object.

\*

\* @param language Locale object

\* @return a corresponding ThriftLanguage object, or UNKNOWN if there's no corresponding one.

\*/

public static ThriftLanguage getThriftLanguageOf(Locale language) {

Preconditions.checkNotNull(language);

ThriftLanguage thriftLang = THRIFT\_LANGUAGES.get(language);

return thriftLang == null ? ThriftLanguage.UNKNOWN : thriftLang;

}

/\*\*

\* Returns a ThriftLanguage object which corresponds to a given language code.

\*

\* @param languageCode BCP-47 language code

\* @return a corresponding ThriftLanguage object, or UNKNOWN if there's no corresponding one.

\*/

public static ThriftLanguage getThriftLanguageOf(String languageCode) {

Preconditions.checkNotNull(languageCode);

ThriftLanguage thriftLang = THRIFT\_LANGUAGES.get(LocaleUtil.getLocaleOf(languageCode));

return thriftLang == null ? ThriftLanguage.UNKNOWN : thriftLang;

}

/\*\*

\* Returns a ThriftLanguage object which corresponds to a given int value.

\* If value is not valid, returns ThriftLanguage.UNKNOWN

\* @param value value of language

\* @return a corresponding ThriftLanguage object

\*/

public static ThriftLanguage safeFindByValue(int value) {

ThriftLanguage thriftLang = ThriftLanguage.findByValue(value);

return thriftLang == null ? ThriftLanguage.UNKNOWN : thriftLang;

}

/\*\*

\* Returns the language code which corresponds to a given ThriftLanguage.

\*

\* Note that multiple ThriftLanguage entries can return the same language code.

\*

\* @param thriftLang ThriftLanguage object

\* @return Corresponding language or null if thriftLang is null.

\*/

@Nullable

public static String getLanguageCodeOf(@Nullable ThriftLanguage thriftLang) {

if (thriftLang == null) {

return null;

}

return ThriftLanguageUtil.getLocaleOf(thriftLang).getLanguage();

}

}