package com.twitter.visibility.generators

import com.twitter.visibility.common.actions.LocalizedMessage

import com.twitter.visibility.common.actions.MessageLink

import com.twitter.visibility.results.translation.Translator

import com.twitter.visibility.results.richtext.EpitaphToRichText

import com.twitter.visibility.results.translation.Resource

import com.twitter.visibility.results.translation.LearnMoreLink

import com.twitter.visibility.rules.Epitaph

import com.twitter.visibility.results.richtext.EpitaphToRichText.Copy

object EpitaphToLocalizedMessage {

def apply(

epitaph: Epitaph,

languageTag: String,

): LocalizedMessage = {

val copy =

EpitaphToRichText.EpitaphToPolicyMap.getOrElse(epitaph, EpitaphToRichText.FallbackPolicy)

val text = Translator.translate(

copy.resource,

languageTag

)

localizeWithCopyAndText(copy, languageTag, text)

}

def apply(

epitaph: Epitaph,

languageTag: String,

applicableCountries: Seq[String],

): LocalizedMessage = {

val copy =

EpitaphToRichText.EpitaphToPolicyMap.getOrElse(epitaph, EpitaphToRichText.FallbackPolicy)

val text = Translator.translateWithSimplePlaceholderReplacement(

copy.resource,

languageTag,

Map((Resource.ApplicableCountriesPlaceholder -> applicableCountries.mkString(", ")))

)

localizeWithCopyAndText(copy, languageTag, text)

}

private def localizeWithCopyAndText(

copy: Copy,

languageTag: String,

text: String

): LocalizedMessage = {

val learnMore = Translator.translate(LearnMoreLink, languageTag)

val links = copy.additionalLinks match {

case links if links.nonEmpty =>

MessageLink(Resource.LearnMorePlaceholder, learnMore, copy.link) +:

links.map {

case EpitaphToRichText.Link(placeholder, copyResource, link) =>

val copyText = Translator.translate(copyResource, languageTag)

MessageLink(placeholder, copyText, link)

}

case \_ =>

Seq(

MessageLink(

key = Resource.LearnMorePlaceholder,

displayText = learnMore,

uri = copy.link))

}

LocalizedMessage(message = text, language = languageTag, links = links)

}

}