package com.twitter.visibility.generators

import com.twitter.decider.Decider

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.visibility.builder.VisibilityResult

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

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

import com.twitter.visibility.configapi.configs.VisibilityDeciderGates

import com.twitter.visibility.results.richtext.PublicInterestReasonToRichText

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

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

import com.twitter.visibility.results.translation.SafetyResultReasonToResource

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

import com.twitter.visibility.rules.EmergencyDynamicInterstitial

import com.twitter.visibility.rules.Interstitial

import com.twitter.visibility.rules.InterstitialLimitedEngagements

import com.twitter.visibility.rules.PublicInterest

import com.twitter.visibility.rules.Reason

import com.twitter.visibility.rules.TweetInterstitial

object LocalizedInterstitialGenerator {

def apply(

visibilityDecider: Decider,

baseStatsReceiver: StatsReceiver,

): LocalizedInterstitialGenerator = {

new LocalizedInterstitialGenerator(visibilityDecider, baseStatsReceiver)

}

}

class LocalizedInterstitialGenerator private (

val visibilityDecider: Decider,

val baseStatsReceiver: StatsReceiver) {

private val visibilityDeciderGates = VisibilityDeciderGates(visibilityDecider)

private val localizationStatsReceiver = baseStatsReceiver.scope("interstitial\_localization")

private val publicInterestInterstitialStats =

localizationStatsReceiver.scope("public\_interest\_copy")

private val emergencyDynamicInterstitialStats =

localizationStatsReceiver.scope("emergency\_dynamic\_copy")

private val regularInterstitialStats = localizationStatsReceiver.scope("interstitial\_copy")

def apply(visibilityResult: VisibilityResult, languageTag: String): VisibilityResult = {

if (!visibilityDeciderGates.enableLocalizedInterstitialGenerator()) {

return visibilityResult

}

visibilityResult.verdict match {

case ipi: InterstitialLimitedEngagements if PublicInterest.Reasons.contains(ipi.reason) =>

visibilityResult.copy(

verdict = ipi.copy(

localizedMessage = Some(localizePublicInterestCopyInResult(ipi, languageTag))

))

case edi: EmergencyDynamicInterstitial =>

visibilityResult.copy(

verdict = EmergencyDynamicInterstitial(

edi.copy,

edi.linkOpt,

Some(localizeEmergencyDynamicCopyInResult(edi, languageTag))

))

case interstitial: Interstitial =>

visibilityResult.copy(

verdict = interstitial.copy(

localizedMessage = localizeInterstitialCopyInResult(interstitial, languageTag)

))

case tweetInterstitial: TweetInterstitial if tweetInterstitial.interstitial.isDefined =>

tweetInterstitial.interstitial.get match {

case ipi: InterstitialLimitedEngagements if PublicInterest.Reasons.contains(ipi.reason) =>

visibilityResult.copy(

verdict = tweetInterstitial.copy(

interstitial = Some(

ipi.copy(

localizedMessage = Some(localizePublicInterestCopyInResult(ipi, languageTag))

))

))

case edi: EmergencyDynamicInterstitial =>

visibilityResult.copy(

verdict = tweetInterstitial.copy(

interstitial = Some(

EmergencyDynamicInterstitial(

edi.copy,

edi.linkOpt,

Some(localizeEmergencyDynamicCopyInResult(edi, languageTag))

))

))

case interstitial: Interstitial =>

visibilityResult.copy(

verdict = tweetInterstitial.copy(

interstitial = Some(

interstitial.copy(

localizedMessage = localizeInterstitialCopyInResult(interstitial, languageTag)

))

))

case \_ => visibilityResult

}

case \_ => visibilityResult

}

}

private def localizeEmergencyDynamicCopyInResult(

edi: EmergencyDynamicInterstitial,

languageTag: String

): LocalizedMessage = {

val text = edi.linkOpt

.map(\_ => s"${edi.copy} {${Resource.LearnMorePlaceholder}}")

.getOrElse(edi.copy)

val messageLinks = edi.linkOpt

.map { link =>

val learnMoreText = Translator.translate(LearnMoreLink, languageTag)

Seq(MessageLink(Resource.LearnMorePlaceholder, learnMoreText, link))

}.getOrElse(Seq.empty)

emergencyDynamicInterstitialStats.counter("localized").incr()

LocalizedMessage(text, languageTag, messageLinks)

}

private def localizePublicInterestCopyInResult(

ipi: InterstitialLimitedEngagements,

languageTag: String

): LocalizedMessage = {

val safetyResultReason = PublicInterest.ReasonToSafetyResultReason(ipi.reason)

val text = Translator.translate(

SafetyResultReasonToResource.resource(safetyResultReason),

languageTag,

)

val learnMoreLink = PublicInterestReasonToRichText.toLearnMoreLink(safetyResultReason)

val learnMoreText = Translator.translate(LearnMoreLink, languageTag)

val messageLinks = Seq(MessageLink(Resource.LearnMorePlaceholder, learnMoreText, learnMoreLink))

publicInterestInterstitialStats.counter("localized").incr()

LocalizedMessage(text, languageTag, messageLinks)

}

private def localizeInterstitialCopyInResult(

interstitial: Interstitial,

languageTag: String

): Option[LocalizedMessage] = {

val localizedMessageOpt = Reason

.toInterstitialReason(interstitial.reason)

.flatMap(InterstitialReasonToLocalizedMessage(\_, languageTag))

if (localizedMessageOpt.isDefined) {

regularInterstitialStats.counter("localized").incr()

localizedMessageOpt

} else {

regularInterstitialStats.counter("empty").incr()

None

}

}

}