package com.twitter.visibility.generators

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.servo.util.MemoizingStatsReceiver

import com.twitter.visibility.builder.VisibilityResult

import com.twitter.visibility.common.actions.TombstoneReason

import com.twitter.visibility.configapi.VisibilityParams

import com.twitter.visibility.rules.Epitaph

import com.twitter.visibility.rules.LocalizedTombstone

import com.twitter.visibility.rules.Tombstone

object TombstoneGenerator {

def apply(

visibilityParams: VisibilityParams,

countryNameGenerator: CountryNameGenerator,

statsReceiver: StatsReceiver

): TombstoneGenerator = {

new TombstoneGenerator(visibilityParams, countryNameGenerator, statsReceiver)

}

}

class TombstoneGenerator(

paramsFactory: VisibilityParams,

countryNameGenerator: CountryNameGenerator,

baseStatsReceiver: StatsReceiver) {

private[this] val statsReceiver = new MemoizingStatsReceiver(

baseStatsReceiver.scope("tombstone\_generator"))

private[this] val deletedReceiver = statsReceiver.scope("deleted\_state")

private[this] val authorStateReceiver = statsReceiver.scope("tweet\_author\_state")

private[this] val visResultReceiver = statsReceiver.scope("visibility\_result")

def apply(

result: VisibilityResult,

language: String

): VisibilityResult = {

result.verdict match {

case tombstone: Tombstone =>

val epitaph = tombstone.epitaph

visResultReceiver.scope("tombstone").counter(epitaph.name.toLowerCase())

val overriddenLanguage = epitaph match {

case Epitaph.LegalDemandsWithheldMedia | Epitaph.LocalLawsWithheldMedia => "en"

case \_ => language

}

tombstone.applicableCountryCodes match {

case Some(countryCodes) => {

val countryNames = countryCodes.map(countryNameGenerator.getCountryName(\_))

result.copy(verdict = LocalizedTombstone(

reason = epitaphToTombstoneReason(epitaph),

message = EpitaphToLocalizedMessage(epitaph, overriddenLanguage, countryNames)))

}

case \_ => {

result.copy(verdict = LocalizedTombstone(

reason = epitaphToTombstoneReason(epitaph),

message = EpitaphToLocalizedMessage(epitaph, overriddenLanguage)))

}

}

case \_ =>

result

}

}

private def epitaphToTombstoneReason(epitaph: Epitaph): TombstoneReason = {

epitaph match {

case Epitaph.Deleted => TombstoneReason.Deleted

case Epitaph.Bounced => TombstoneReason.Bounced

case Epitaph.BounceDeleted => TombstoneReason.BounceDeleted

case Epitaph.Protected => TombstoneReason.ProtectedAuthor

case Epitaph.Suspended => TombstoneReason.SuspendedAuthor

case Epitaph.BlockedBy => TombstoneReason.AuthorBlocksViewer

case Epitaph.SuperFollowsContent => TombstoneReason.ExclusiveTweet

case Epitaph.Underage => TombstoneReason.NsfwViewerIsUnderage

case Epitaph.NoStatedAge => TombstoneReason.NsfwViewerHasNoStatedAge

case Epitaph.LoggedOutAge => TombstoneReason.NsfwLoggedOut

case Epitaph.Deactivated => TombstoneReason.DeactivatedAuthor

case Epitaph.CommunityTweetHidden => TombstoneReason.CommunityTweetHidden

case Epitaph.CommunityTweetCommunityIsSuspended =>

TombstoneReason.CommunityTweetCommunityIsSuspended

case Epitaph.DevelopmentOnly => TombstoneReason.DevelopmentOnly

case Epitaph.AdultMedia => TombstoneReason.AdultMedia

case Epitaph.ViolentMedia => TombstoneReason.ViolentMedia

case Epitaph.OtherSensitiveMedia => TombstoneReason.OtherSensitiveMedia

case Epitaph.DmcaWithheldMedia => TombstoneReason.DmcaWithheldMedia

case Epitaph.LegalDemandsWithheldMedia => TombstoneReason.LegalDemandsWithheldMedia

case Epitaph.LocalLawsWithheldMedia => TombstoneReason.LocalLawsWithheldMedia

case Epitaph.ToxicReplyFiltered => TombstoneReason.ReplyFiltered

case \_ => TombstoneReason.Unspecified

}

}

}