package com.twitter.visibility.builder.media

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.mediaservices.media\_util.GenericMediaKey

import com.twitter.stitch.Stitch

import com.twitter.visibility.builder.FeatureMapBuilder

import com.twitter.visibility.common.MediaSafetyLabelMapSource

import com.twitter.visibility.features.MediaSafetyLabels

import com.twitter.visibility.models.MediaSafetyLabel

import com.twitter.visibility.models.MediaSafetyLabelType

import com.twitter.visibility.models.SafetyLabel

class MediaFeatures(

mediaSafetyLabelMap: StratoMediaLabelMaps,

statsReceiver: StatsReceiver) {

private[this] val scopedStatsReceiver = statsReceiver.scope("media\_features")

private[this] val requests =

scopedStatsReceiver

.counter("requests")

private[this] val mediaSafetyLabelsStats =

scopedStatsReceiver

.scope(MediaSafetyLabels.name)

.counter("requests")

private[this] val nonEmptyMediaStats = scopedStatsReceiver.scope("non\_empty\_media")

private[this] val nonEmptyMediaRequests = nonEmptyMediaStats.counter("requests")

private[this] val nonEmptyMediaKeysCount = nonEmptyMediaStats.counter("keys")

private[this] val nonEmptyMediaKeysLength = nonEmptyMediaStats.stat("keys\_length")

def forMediaKeys(

mediaKeys: Seq[GenericMediaKey],

): FeatureMapBuilder => FeatureMapBuilder = {

requests.incr()

nonEmptyMediaKeysCount.incr(mediaKeys.size)

mediaSafetyLabelsStats.incr()

if (mediaKeys.nonEmpty) {

nonEmptyMediaRequests.incr()

nonEmptyMediaKeysLength.add(mediaKeys.size)

}

\_.withFeature(MediaSafetyLabels, mediaSafetyLabelMap.forGenericMediaKeys(mediaKeys))

}

def forGenericMediaKey(

genericMediaKey: GenericMediaKey

): FeatureMapBuilder => FeatureMapBuilder = {

requests.incr()

nonEmptyMediaKeysCount.incr()

mediaSafetyLabelsStats.incr()

nonEmptyMediaRequests.incr()

nonEmptyMediaKeysLength.add(1L)

\_.withFeature(MediaSafetyLabels, mediaSafetyLabelMap.forGenericMediaKey(genericMediaKey))

}

}

class StratoMediaLabelMaps(source: MediaSafetyLabelMapSource) {

def forGenericMediaKeys(

mediaKeys: Seq[GenericMediaKey],

): Stitch[Seq[MediaSafetyLabel]] = {

Stitch

.collect(

mediaKeys

.map(getFilteredSafetyLabels)

).map(\_.flatten)

}

def forGenericMediaKey(

genericMediaKey: GenericMediaKey

): Stitch[Seq[MediaSafetyLabel]] = {

getFilteredSafetyLabels(genericMediaKey)

}

private def getFilteredSafetyLabels(

genericMediaKey: GenericMediaKey,

): Stitch[Seq[MediaSafetyLabel]] =

source

.fetch(genericMediaKey).map(\_.flatMap(\_.labels.map { stratoSafetyLabelMap =>

stratoSafetyLabelMap

.map(label =>

MediaSafetyLabel(

MediaSafetyLabelType.fromThrift(label.\_1),

SafetyLabel.fromThrift(label.\_2)))

}).toSeq.flatten)

}