package com.twitter.tweetypie

package service

package observer

import com.twitter.tweetypie.thriftscala.StoredTweetError

import com.twitter.tweetypie.thriftscala.StoredTweetInfo

import com.twitter.tweetypie.thriftscala.StoredTweetState.BounceDeleted

import com.twitter.tweetypie.thriftscala.StoredTweetState.ForceAdded

import com.twitter.tweetypie.thriftscala.StoredTweetState.HardDeleted

import com.twitter.tweetypie.thriftscala.StoredTweetState.NotFound

import com.twitter.tweetypie.thriftscala.StoredTweetState.SoftDeleted

import com.twitter.tweetypie.thriftscala.StoredTweetState.Undeleted

import com.twitter.tweetypie.thriftscala.StoredTweetState.UnknownUnionField

private[service] trait StoredTweetsObserver {

protected def observeStoredTweets(

storedTweets: Seq[StoredTweetInfo],

stats: StatsReceiver

): Unit = {

val stateScope = stats.scope("state")

val errorScope = stats.scope("error")

val sizeCounter = stats.counter("count")

sizeCounter.incr(storedTweets.size)

val returnedStatesCount = storedTweets

.groupBy(\_.storedTweetState match {

case None => "found"

case Some(\_: HardDeleted) => "hard\_deleted"

case Some(\_: SoftDeleted) => "soft\_deleted"

case Some(\_: BounceDeleted) => "bounce\_deleted"

case Some(\_: Undeleted) => "undeleted"

case Some(\_: ForceAdded) => "force\_added"

case Some(\_: NotFound) => "not\_found"

case Some(\_: UnknownUnionField) => "unknown"

})

.mapValues(\_.size)

returnedStatesCount.foreach {

case (state, count) => stateScope.counter(state).incr(count)

}

val returnedErrorsCount = storedTweets

.foldLeft(Seq[StoredTweetError]()) { (errors, storedTweetInfo) =>

errors ++ storedTweetInfo.errors

}

.groupBy(\_.name)

.mapValues(\_.size)

returnedErrorsCount.foreach {

case (error, count) => errorScope.counter(error).incr(count)

}

}

}