package com.twitter.tweetypie

package service

package observer

import com.twitter.tweetypie.thriftscala.GetStoredTweetsRequest

import com.twitter.tweetypie.thriftscala.GetStoredTweetsResult

private[service] object GetStoredTweetsObserver extends StoredTweetsObserver {

type Type = ObserveExchange[GetStoredTweetsRequest, Seq[GetStoredTweetsResult]]

def observeRequest(stats: StatsReceiver): Effect[GetStoredTweetsRequest] = {

val requestSizeStat = stats.stat("request\_size")

val optionsScope = stats.scope("options")

val bypassVisibilityFilteringCounter = optionsScope.counter("bypass\_visibility\_filtering")

val forUserIdCounter = optionsScope.counter("for\_user\_id")

val additionalFieldsScope = optionsScope.scope("additional\_fields")

Effect { request =>

requestSizeStat.add(request.tweetIds.size)

if (request.options.isDefined) {

val options = request.options.get

if (options.bypassVisibilityFiltering) bypassVisibilityFilteringCounter.incr()

if (options.forUserId.isDefined) forUserIdCounter.incr()

options.additionalFieldIds.foreach { id =>

additionalFieldsScope.counter(id.toString).incr()

}

}

}

}

def observeResult(stats: StatsReceiver): Effect[Seq[GetStoredTweetsResult]] = {

val resultScope = stats.scope("result")

Effect { result =>

observeStoredTweets(result.map(\_.storedTweet), resultScope)

}

}

def observeExchange(stats: StatsReceiver): Effect[Type] = {

val resultStateStats = ResultStateStats(stats)

Effect {

case (request, response) =>

response match {

case Return(\_) => resultStateStats.success(request.tweetIds.size)

case Throw(\_) => resultStateStats.failed(request.tweetIds.size)

}

}

}

}