package com.twitter.servo.request

import com.twitter.finagle.stats.NullStatsReceiver

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.util.Future

object ClientRequestObserver {

private[request] val noClientIdKey = "no\_client\_id"

}

/\*\*

\* Provides per-request stats based on Finagle ClientId.

\*

\* @param statsReceiver the StatsReceiver used for counting

\* @param observeAuthorizationAttempts: if true (the default), observe all attempts. If false,

\* only failures (unauthorized attempts) are observed.

\*/

class ClientRequestObserver(

statsReceiver: StatsReceiver,

observeAuthorizationAttempts: Boolean = true)

extends ((String, Option[Seq[String]]) => Future[Unit]) {

import ClientRequestObserver.noClientIdKey

protected[this] val scopedReceiver = statsReceiver.scope("client\_request")

protected[this] val unauthorizedReceiver = scopedReceiver.scope("unauthorized")

protected[this] val unauthorizedCounter = scopedReceiver.counter("unauthorized")

/\*\*

\* @param methodName the name of the Service method being called

\* @param clientIdScopesOpt optional sequence of scope strings representing the

\* originating request's ClientId

\*/

override def apply(methodName: String, clientIdScopesOpt: Option[Seq[String]]): Future[Unit] = {

if (observeAuthorizationAttempts) {

scopedReceiver.counter(methodName).incr()

clientIdScopesOpt match {

case Some(clientIdScopes) =>

scopedReceiver.scope(methodName).counter(clientIdScopes: \_\*).incr()

case None =>

scopedReceiver.scope(methodName).counter(noClientIdKey).incr()

}

}

Future.Done

}

/\*\*

\* Increments a counter for unauthorized requests.

\*/

def unauthorized(methodName: String, clientIdStr: String): Unit = {

unauthorizedCounter.incr()

unauthorizedReceiver.scope(methodName).counter(clientIdStr).incr()

}

def authorized(methodName: String, clientIdStr: String): Unit = {}

}

object NullClientRequestObserver extends ClientRequestObserver(NullStatsReceiver)