package com.twitter.tweetypie

package config

import com.twitter.io.Buf

import com.twitter.finagle.{Service, SimpleFilter}

import com.twitter.finagle.memcached.protocol.\_

class MemcacheExceptionLoggingFilter extends SimpleFilter[Command, Response] {

// Using a custom logger name so that we can target logging rules specifically

// for memcache excpetion logging.

val logger: Logger = Logger(getClass)

def apply(command: Command, service: Service[Command, Response]): Future[Response] = {

service(command).respond {

case Return(Error(e)) =>

log(command, e)

case Return(ValuesAndErrors(\_, errors)) if errors.nonEmpty =>

errors.foreach {

case (Buf.Utf8(keyStr), e) =>

log(command.name, keyStr, e)

}

case Throw(e) =>

log(command, e)

case \_ =>

}

}

private def log(command: Command, e: Throwable): Unit = {

log(command.name, getKey(command), e)

}

private def log(commandName: String, keyStr: String, e: Throwable): Unit = {

logger.debug(

s"CACHE\_EXCEPTION command: ${commandName} key: ${keyStr} exception: ${e.getClass.getName}",

e,

)

}

private def getKey(command: Command): String = command match {

case Get(keys) => toKeyStr(keys)

case Gets(keys) => toKeyStr(keys)

case Set(Buf.Utf8(key), \_, \_, \_) => key

case Add(Buf.Utf8(key), \_, \_, \_) => key

case Cas(Buf.Utf8(key), \_, \_, \_, \_) => key

case Delete(Buf.Utf8(key)) => key

case Replace(Buf.Utf8(key), \_, \_, \_) => key

case Append(Buf.Utf8(key), \_, \_, \_) => key

case Prepend(Buf.Utf8(key), \_, \_, \_) => key

case Incr(Buf.Utf8(key), \_) => key

case Decr(Buf.Utf8(key), \_) => key

case Stats(keys) => toKeyStr(keys)

case Quit() => "quit"

case Upsert(Buf.Utf8(key), \_, \_, \_, \_) => key

case Getv(keys) => toKeyStr(keys)

}

private def toKeyStr(keys: Seq[Buf]): String =

keys.map { case Buf.Utf8(key) => key }.mkString(",")

}