package com.twitter.tweetypie.serverutil.logcachewrites

import com.twitter.servo.cache.Checksum

import com.twitter.servo.cache.CacheWrapper

import com.twitter.util.Future

import com.twitter.util.logging.Logger

import scala.util.control.NonFatal

trait WriteLoggingCache[K, V] extends CacheWrapper[K, V] {

// Use getClass so we can see which implementation is actually failing.

private[this] lazy val logFailureLogger = Logger(getClass)

def selectKey(k: K): Boolean

def select(k: K, v: V): Boolean

def log(action: String, k: K, v: Option[V]): Unit

def safeLog(action: String, k: K, v: Option[V]): Unit =

try {

log(action, k, v)

} catch {

case NonFatal(e) =>

// The exception occurred in logging, and we don't want to fail the

// request with the logging failure if this happens, so log it and carry

// on.

logFailureLogger.error("Logging cache write", e)

}

override def add(k: K, v: V): Future[Boolean] =

// Call the selection function before doing the work. Since it's highly

// likely that the Future will succeed, it's cheaper to call the function

// before we make the call so that we can avoid creating the callback and

// attaching it to the Future if we would not log.

if (select(k, v)) {

underlyingCache.add(k, v).onSuccess(r => if (r) safeLog("add", k, Some(v)))

} else {

underlyingCache.add(k, v)

}

override def checkAndSet(k: K, v: V, checksum: Checksum): Future[Boolean] =

if (select(k, v)) {

underlyingCache.checkAndSet(k, v, checksum).onSuccess(r => if (r) safeLog("cas", k, Some(v)))

} else {

underlyingCache.checkAndSet(k, v, checksum)

}

override def set(k: K, v: V): Future[Unit] =

if (select(k, v)) {

underlyingCache.set(k, v).onSuccess(\_ => safeLog("set", k, Some(v)))

} else {

underlyingCache.set(k, v)

}

override def replace(k: K, v: V): Future[Boolean] =

if (select(k, v)) {

underlyingCache.replace(k, v).onSuccess(r => if (r) safeLog("replace", k, Some(v)))

} else {

underlyingCache.replace(k, v)

}

override def delete(k: K): Future[Boolean] =

if (selectKey(k)) {

underlyingCache.delete(k).onSuccess(r => if (r) safeLog("delete", k, None))

} else {

underlyingCache.delete(k)

}

}