package com.twitter.tweetypie

package service

import com.twitter.finagle.IndividualRequestTimeoutException

import com.twitter.servo.exception.thriftscala.\_

import com.twitter.tweetypie.core.OverCapacity

import com.twitter.tweetypie.core.RateLimited

import com.twitter.tweetypie.core.TweetHydrationError

import com.twitter.tweetypie.core.UpstreamFailure

import com.twitter.tweetypie.thriftscala.\_

import com.twitter.util.TimeoutException

object RescueExceptions {

private val log = Logger("com.twitter.tweetypie.service.TweetService")

/\*\*

\* rescue to servo exceptions

\*/

def rescueToServoFailure(

name: String,

clientId: String

): PartialFunction[Throwable, Future[Nothing]] = {

translateToServoFailure(formatError(name, clientId, \_)).andThen(Future.exception)

}

private def translateToServoFailure(

toMsg: String => String

): PartialFunction[Throwable, Throwable] = {

case e: AccessDenied if suspendedOrDeactivated(e) =>

e.copy(message = toMsg(e.message))

case e: ClientError =>

e.copy(message = toMsg(e.message))

case e: UnauthorizedException =>

ClientError(ClientErrorCause.Unauthorized, toMsg(e.msg))

case e: AccessDenied =>

ClientError(ClientErrorCause.Unauthorized, toMsg(e.message))

case e: RateLimited =>

ClientError(ClientErrorCause.RateLimited, toMsg(e.message))

case e: ServerError =>

e.copy(message = toMsg(e.message))

case e: TimeoutException =>

ServerError(ServerErrorCause.RequestTimeout, toMsg(e.toString))

case e: IndividualRequestTimeoutException =>

ServerError(ServerErrorCause.RequestTimeout, toMsg(e.toString))

case e: UpstreamFailure =>

ServerError(ServerErrorCause.DependencyError, toMsg(e.toString))

case e: OverCapacity =>

ServerError(ServerErrorCause.ServiceUnavailable, toMsg(e.message))

case e: TweetHydrationError =>

ServerError(ServerErrorCause.DependencyError, toMsg(e.toString))

case e =>

log.warn("caught unexpected exception", e)

ServerError(ServerErrorCause.InternalServerError, toMsg(e.toString))

}

private def suspendedOrDeactivated(e: AccessDenied): Boolean =

e.errorCause.exists { c =>

c == AccessDeniedCause.UserDeactivated || c == AccessDeniedCause.UserSuspended

}

private def formatError(name: String, clientId: String, msg: String): String =

s"($clientId, $name) $msg"

}