package com.twitter.tweetypie

package service

import com.twitter.bijection.scrooge.BinaryScalaCodec

import com.twitter.coreservices.failed\_task.writer.FailedTaskWriter

import com.twitter.scrooge.ThriftException

import com.twitter.scrooge.ThriftStruct

import com.twitter.scrooge.ThriftStructCodec

import com.twitter.tweetypie.serverutil.BoringStackTrace

import com.twitter.tweetypie.thriftscala.\_

import scala.util.control.NoStackTrace

object FailureLoggingTweetService {

/\*\*

\* Defines the universe of exception types for which we should scribe

\* the failure.

\*/

private def shouldWrite(t: Throwable): Boolean =

t match {

case \_: ThriftException => true

case \_: PostTweetFailure => true

case \_ => !BoringStackTrace.isBoring(t)

}

/\*\*

\* Holds failure information from a failing PostTweetResult.

\*

\* FailedTaskWriter logs an exception with the failed request, so we

\* need to package up any failure that we want to log into an

\* exception.

\*/

private class PostTweetFailure(state: TweetCreateState, reason: Option[String])

extends Exception

with NoStackTrace {

override def toString: String = s"PostTweetFailure($state, $reason)"

}

}

/\*\*

\* Wraps a tweet service with scribing of failed requests in order to

\* enable analysis of failures for diagnosing problems.

\*/

class FailureLoggingTweetService(

failedTaskWriter: FailedTaskWriter[Array[Byte]],

protected val underlying: ThriftTweetService)

extends TweetServiceProxy {

import FailureLoggingTweetService.\_

private[this] object writers {

private[this] def writer[T <: ThriftStruct](

name: String,

codec: ThriftStructCodec[T]

): (T, Throwable) => Future[Unit] = {

val taskWriter = failedTaskWriter(name, BinaryScalaCodec(codec).apply)

(t, exc) =>

Future.when(shouldWrite(exc)) {

taskWriter.writeFailure(t, exc)

}

}

val postTweet: (PostTweetRequest, Throwable) => Future[Unit] =

writer("post\_tweet", PostTweetRequest)

}

override def postTweet(request: PostTweetRequest): Future[PostTweetResult] =

underlying.postTweet(request).respond {

// Log requests for states other than OK to enable debugging creation failures

case Return(res) if res.state != TweetCreateState.Ok =>

writers.postTweet(request, new PostTweetFailure(res.state, res.failureReason))

case Throw(exc) =>

writers.postTweet(request, exc)

case \_ =>

}

}