package com.twitter.tweetypie.serverutil

import com.twitter.finagle.ChannelException

import com.twitter.finagle.TimeoutException

import com.twitter.scrooge.ThriftException

import java.net.SocketException

import java.nio.channels.CancelledKeyException

import java.nio.channels.ClosedChannelException

import java.util.concurrent.CancellationException

import java.util.concurrent.{TimeoutException => JTimeoutException}

import org.apache.thrift.TApplicationException

import scala.util.control.NoStackTrace

object BoringStackTrace {

/\*\*

\* These exceptions are boring because they are expected to

\* occasionally (or even regularly) happen during normal operation

\* of the service. The intention is to make it easier to debug

\* problems by making interesting exceptions easier to see.

\*

\* The best way to mark an exception as boring is to extend from

\* NoStackTrace, since that is a good indication that we don't care

\* about the details.

\*/

def isBoring(t: Throwable): Boolean =

t match {

case \_: NoStackTrace => true

case \_: TimeoutException => true

case \_: CancellationException => true

case \_: JTimeoutException => true

case \_: ChannelException => true

case \_: SocketException => true

case \_: ClosedChannelException => true

case \_: CancelledKeyException => true

case \_: ThriftException => true

// DeadlineExceededExceptions are propagated as:

// org.apache.thrift.TApplicationException: Internal error processing issue3: 'com.twitter.finagle.service.DeadlineFilter$DeadlineExceededException: exceeded request deadline of 100.milliseconds by 4.milliseconds. Deadline expired at 2020-08-27 17:07:46 +0000 and now it is 2020-08-27 17:07:46 +0000.'

case e: TApplicationException =>

e.getMessage != null && e.getMessage.contains("DeadlineExceededException")

case \_ => false

}

}