package com.twitter.tweetypie.util.logging

import ch.qos.logback.classic.spi.ILoggingEvent

import ch.qos.logback.classic.spi.ThrowableProxy

import ch.qos.logback.core.filter.Filter

import ch.qos.logback.core.spi.FilterReply

import com.twitter.tweetypie.serverutil.ExceptionCounter.isAlertable

/\*\*

\* This class is currently being used by logback to log alertable exceptions to a seperate file.

\*

\* Filters do not change the log levels of individual loggers. Filters filter out specific messages

\* for specific appenders. This allows us to have a log file with lots of information you will

\* mostly not need and a log file with only important information. This type of filtering cannot be

\* accomplished by changing the log levels of loggers, because the logger levels are global. We want

\* to change the semantics for specific destinations (appenders).

\*/

class AlertableExceptionLoggingFilter extends Filter[ILoggingEvent] {

private[this] val IgnorableLoggers: Set[String] =

Set(

"com.github.benmanes.caffeine.cache.BoundedLocalCache",

"abdecider",

"org.apache.kafka.common.network.SaslChannelBuilder",

"com.twitter.finagle.netty4.channel.ChannelStatsHandler$"

)

def include(proxy: ThrowableProxy, event: ILoggingEvent): Boolean =

isAlertable(proxy.getThrowable()) && !IgnorableLoggers(event.getLoggerName)

override def decide(event: ILoggingEvent): FilterReply =

if (!isStarted) {

FilterReply.NEUTRAL

} else {

event.getThrowableProxy() match {

case proxy: ThrowableProxy if include(proxy, event) =>

FilterReply.NEUTRAL

case \_ =>

FilterReply.DENY

}

}

}