package com.twitter.search.earlybird.exception;

import com.google.common.annotations.VisibleForTesting;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.slf4j.Marker;

import org.slf4j.MarkerFactory;

import com.twitter.search.common.config.Config;

import com.twitter.search.common.metrics.SearchCounter;

import com.twitter.search.earlybird.EarlybirdStatus;

/\*\*

\* Used for handling exceptions considered critical.

\*

\* When you handle an exception with this class, two things might happen.

\* 1. If earlybirds are still starting, we'll shut them down.

\* 2. If earlybirds have started, we'll increment a counter that will cause alerts.

\*

\* If you want to verify that your code handles exceptions as you expect, you can use the

\* helper class ExceptionCauser.

\*/

public class CriticalExceptionHandler {

private static final Logger LOG = LoggerFactory.getLogger(CriticalExceptionHandler.class);

private static final Marker FATAL = MarkerFactory.getMarker("FATAL");

// This stat should remain at 0 during normal operations.

// This stat being non-zero should trigger alerts.

public static final SearchCounter CRITICAL\_EXCEPTION\_COUNT =

SearchCounter.export("fatal\_exception\_count");

public static final SearchCounter UNSAFE\_MEMORY\_ACCESS =

SearchCounter.export("unsafe\_memory\_access");

private Runnable shutdownHook;

public void setShutdownHook(Runnable shutdownHook) {

this.shutdownHook = shutdownHook;

}

/\*\*

\* Handle a critical exception.

\*

\* @param thrower Instance of the class where the exception was thrown.

\* @param thrown The exception.

\*/

public void handle(Object thrower, Throwable thrown) {

if (thrown == null) {

return;

}

try {

handleFatalException(thrower, thrown);

} catch (Throwable e) {

LOG.error("Unexpected exception in EarlybirdExceptionHandler.handle() while handling an "

+ "unexpected exception from " + thrower.getClass(), e);

}

}

@VisibleForTesting

boolean shouldIncrementFatalExceptionCounter(Throwable thrown) {

// See D212952

// We don't want to get pages when this happens.

for (Throwable t = thrown; t != null; t = t.getCause()) {

if (t instanceof InternalError && t.getMessage() != null

&& t.getMessage().contains("unsafe memory access operation")) {

// Don't treat InternalError caused by unsafe memory access operation which is usually

// triggered by SIGBUS for accessing a corrupted memory block.

UNSAFE\_MEMORY\_ACCESS.increment();

return false;

}

}

return true;

}

/\*\*

\* Handle an exception that's considered fatal.

\*

\* @param thrower instance of the class where the exception was thrown.

\* @param thrown The Error or Exception.

\*/

private void handleFatalException(Object thrower, Throwable thrown) {

LOG.error(FATAL, "Fatal exception in " + thrower.getClass() + ":", thrown);

if (shouldIncrementFatalExceptionCounter(thrown)) {

CRITICAL\_EXCEPTION\_COUNT.increment();

}

if (EarlybirdStatus.isStarting()) {

LOG.error(FATAL, "Got fatal exception while starting up, exiting ...");

if (this.shutdownHook != null) {

this.shutdownHook.run();

} else {

LOG.error("earlybirdServer not set, can't shut down.");

}

if (!Config.environmentIsTest()) {

// Sleep for 3 minutes to allow the fatal exception to be caught by observability.

try {

Thread.sleep(3 \* 60 \* 1000);

} catch (InterruptedException e) {

LOG.error(FATAL, "interupted sleep while shutting down.");

}

LOG.info("Terminate JVM.");

//CHECKSTYLE:OFF RegexpSinglelineJava

// See SEARCH-15256

System.exit(-1);

//CHECKSTYLE:ON RegexpSinglelineJava

}

}

}

}