package com.twitter.search.earlybird.util;

import java.io.Closeable;

import java.io.IOException;

import java.util.concurrent.ScheduledExecutorService;

import java.util.concurrent.ScheduledFuture;

import com.twitter.common.util.Clock;

import com.twitter.search.common.concurrent.ScheduledExecutorServiceFactory;

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

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

import com.twitter.search.earlybird.exception.CriticalExceptionHandler;

/\*\*

\* Executes a single periodic task.

\*/

public abstract class OneTaskScheduledExecutorManager

extends ScheduledExecutorManager implements Closeable {

private final ScheduledExecutorTask scheduledTask;

private final PeriodicActionParams periodicActionParams;

public OneTaskScheduledExecutorManager(

ScheduledExecutorServiceFactory executorServiceFactory,

String threadNameFormat,

boolean isDaemon,

PeriodicActionParams periodicActionParams,

ShutdownWaitTimeParams shutdownTiming,

SearchStatsReceiver searchStatsReceiver,

CriticalExceptionHandler criticalExceptionHandler) {

this(executorServiceFactory.build(threadNameFormat, isDaemon), periodicActionParams,

shutdownTiming, searchStatsReceiver, criticalExceptionHandler);

}

public OneTaskScheduledExecutorManager(

ScheduledExecutorService executor,

PeriodicActionParams periodicActionParams,

ShutdownWaitTimeParams shutdownTiming,

SearchStatsReceiver searchStatsReceiver,

CriticalExceptionHandler criticalExceptionHandler) {

this(executor, periodicActionParams, shutdownTiming, searchStatsReceiver, null,

criticalExceptionHandler, Clock.SYSTEM\_CLOCK);

}

public OneTaskScheduledExecutorManager(

ScheduledExecutorService executor,

PeriodicActionParams periodicActionParams,

ShutdownWaitTimeParams shutdownWaitTimeParams,

SearchStatsReceiver searchStatsReceiver,

SearchCounter iterationCounter,

CriticalExceptionHandler criticalExceptionHandler,

Clock clock) {

super(executor, shutdownWaitTimeParams, searchStatsReceiver, iterationCounter,

criticalExceptionHandler, clock);

this.periodicActionParams = periodicActionParams;

this.scheduledTask = new ScheduledExecutorTask(getIterationCounter(), clock) {

@Override

protected void runOneIteration() {

OneTaskScheduledExecutorManager.this.runOneIteration();

}

};

}

/\*\*

\* Schedule the single internally specified task returned by getScheduledTask.

\*/

public ScheduledFuture schedule() {

return this.scheduleNewTask(

this.getScheduledTask(),

this.periodicActionParams

);

}

/\*\*

\* The code that the task executes.

\*/

protected abstract void runOneIteration();

public ScheduledExecutorTask getScheduledTask() {

return scheduledTask;

}

@Override

public void close() throws IOException {

try {

shutdown();

} catch (InterruptedException e) {

throw new IOException(e);

}

}

}