package com.twitter.search.common.search.termination;

import com.google.common.base.Preconditions;

import com.twitter.common.util.Clock;

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

import com.twitter.search.common.search.DocIdTracker;

import com.twitter.search.common.search.EarlyTerminationState;

import com.twitter.search.common.search.TerminationTracker;

/\*\*

\* QueryTimeoutImpl provides a method for early termination of queries based on time.

\*/

public class QueryTimeoutImpl implements QueryTimeout {

private final String clientId;

private final TerminationTracker tracker;

private final Clock clock;

private final SearchRateCounter shouldTerminateCounter;

public QueryTimeoutImpl(String clientId, TerminationTracker tracker, Clock clock) {

this.clientId = Preconditions.checkNotNull(clientId);

this.tracker = Preconditions.checkNotNull(tracker);

this.clock = Preconditions.checkNotNull(clock);

shouldTerminateCounter =

SearchRateCounter.export("query\_timeout\_should\_terminate\_" + clientId);

}

/\*\*

\* Returns true when the clock's time has met or exceeded the tracker's timeout end.

\*/

public boolean shouldExit() {

if (clock.nowMillis() >= tracker.getTimeoutEndTimeWithReservation()) {

tracker.setEarlyTerminationState(EarlyTerminationState.TERMINATED\_TIME\_OUT\_EXCEEDED);

shouldTerminateCounter.increment();

return true;

}

return false;

}

@Override

public void registerDocIdTracker(DocIdTracker docIdTracker) {

tracker.addDocIdTracker(docIdTracker);

}

@Override

public String getClientId() {

return clientId;

}

@Override

public int hashCode() {

return clientId.hashCode() \* 13 + tracker.hashCode();

}

@Override

public boolean equals(Object obj) {

if (!(obj instanceof QueryTimeoutImpl)) {

return false;

}

QueryTimeoutImpl queryTimeout = QueryTimeoutImpl.class.cast(obj);

return clientId.equals(queryTimeout.clientId) && tracker.equals(queryTimeout.tracker);

}

}