package com.twitter.search.earlybird;

import com.twitter.finagle.thrift.ThriftClientRequest;

import com.twitter.search.common.dark.DarkProxy;

import com.twitter.search.earlybird.thrift.EarlybirdService;

import com.twitter.util.Duration;

/\*\*

\* Manages a finagle server underneath, which can be recreated.

\*

\* This class is not thread-safe. It is up to the concrete implementations and their callers to

\* correctly synchronize calls to these methods (for example, to make sure that there is no race

\* condition if startProductionFinagleServer() and stopProductionFinagleServer() are called

\* concurrently from two different threads).

\*/

public interface EarlybirdFinagleServerManager {

/\*\*

\* Determines if the warm up finagle server is currently running

\*/

boolean isWarmUpServerRunning();

/\*\*

\* Starts up the warm up finagle server on the given port.

\*/

void startWarmUpFinagleServer(

EarlybirdService.ServiceIface serviceIface,

String serviceName,

int port);

/\*\*

\* Stops the warm up finagle server, after waiting for at most the given amount of time.

\*/

void stopWarmUpFinagleServer(Duration serverCloseWaitTime) throws InterruptedException;

/\*\*

\* Determines if the production finagle server is currently running.

\*/

boolean isProductionServerRunning();

/\*\*

\* Starts up the production finagle server on the given port.

\*/

void startProductionFinagleServer(

DarkProxy<ThriftClientRequest, byte[]> darkProxy,

EarlybirdService.ServiceIface serviceIface,

String serviceName,

int port);

/\*\*

\* Stops the production finagle server after waiting for at most the given amount of time.

\*/

void stopProductionFinagleServer(Duration serverCloseWaitTime) throws InterruptedException;

}