package com.twitter.tweetypie

package service

import com.twitter.conversions.DurationOps.\_

import com.twitter.finagle.thrift.ClientId

import com.twitter.tweetypie.thriftscala.\_

import com.twitter.util.Await

import scala.util.control.NonFatal

/\*\*

\* Settings for the artificial tweet fetching requests that are sent to warmup the

\* server before authentic requests are processed.

\*/

case class WarmupQueriesSettings(

realTweetRequestCycles: Int = 100,

requestTimeout: Duration = 3.seconds,

clientId: ClientId = ClientId("tweetypie.warmup"),

requestTimeRange: Duration = 10.minutes,

maxConcurrency: Int = 20)

object TweetServiceWarmer {

/\*\*

\* Load info from perspective of TLS test account with short favorites timeline.

\*/

val ForUserId = 3511687034L // @mikestltestact1

}

/\*\*

\* Generates requests to getTweets for the purpose of warming up the code paths used

\* in fetching tweets.

\*/

class TweetServiceWarmer(

warmupSettings: WarmupQueriesSettings,

requestOptions: GetTweetOptions = GetTweetOptions(includePlaces = true,

includeRetweetCount = true, includeReplyCount = true, includeFavoriteCount = true,

includeCards = true, cardsPlatformKey = Some("iPhone-13"), includePerspectivals = true,

includeQuotedTweet = true, forUserId = Some(TweetServiceWarmer.ForUserId)))

extends (ThriftTweetService => Unit) {

import warmupSettings.\_

private val realTweetIds =

Seq(

20L, // just setting up my twttr

456190426412617728L, // protected user tweet

455477977715707904L, // suspended user tweet

440322224407314432L, // ellen oscar selfie

372173241290612736L, // gaga mentions 1d

456965485179838464L, // media tagged tweet

525421442918121473L, // tweet with card

527214829807759360L, // tweet with annotation

472788687571677184L // tweet with quote tweet

)

private val log = Logger(getClass)

/\*\*

\* Executes the warmup queries, waiting for them to complete or until

\* the warmupTimeout occurs.

\*/

def apply(service: ThriftTweetService): Unit = {

val warmupStart = Time.now

log.info("warming up...")

warmup(service)

val warmupDuration = Time.now.since(warmupStart)

log.info("warmup took " + warmupDuration)

}

/\*\*

\* Executes the warmup queries, returning when all responses have completed or timed-out.

\*/

private[this] def warmup(service: ThriftTweetService): Unit =

clientId.asCurrent {

val request = GetTweetsRequest(realTweetIds, options = Some(requestOptions))

val requests = Seq.fill(realTweetRequestCycles)(request)

val requestGroups = requests.grouped(maxConcurrency)

for (requests <- requestGroups) {

val responses = requests.map(service.getTweets(\_))

try {

Await.ready(Future.join(responses), requestTimeout)

} catch {

// Await.ready throws exceptions on timeouts and

// interruptions. This prevents those exceptions from

// bubbling up.

case NonFatal(\_) =>

}

}

}

}