package com.twitter.recos.user\_tweet\_entity\_graph

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.frigate.common.util.StatsUtil

import com.twitter.graphjet.algorithms.RecommendationInfo

import com.twitter.graphjet.algorithms.socialproof.{SocialProofResult => SocialProofJavaResult}

import com.twitter.recos.decider.UserTweetEntityGraphDecider

import com.twitter.recos.util.Stats

import com.twitter.recos.util.Stats.\_

import com.twitter.recos.recos\_common.thriftscala.{SocialProofType => SocialProofThriftType}

import com.twitter.recos.user\_tweet\_entity\_graph.thriftscala.TweetRecommendation

import com.twitter.recos.user\_tweet\_entity\_graph.thriftscala.{

SocialProofRequest => SocialProofThriftRequest

}

import com.twitter.recos.user\_tweet\_entity\_graph.thriftscala.{

SocialProofResponse => SocialProofThriftResponse

}

import com.twitter.servo.request.RequestHandler

import com.twitter.util.Future

import scala.collection.JavaConverters.\_

class TweetSocialProofHandler(

tweetSocialProofRunner: TweetSocialProofRunner,

decider: UserTweetEntityGraphDecider,

statsReceiver: StatsReceiver)

extends RequestHandler[SocialProofThriftRequest, SocialProofThriftResponse] {

private val stats = statsReceiver.scope(this.getClass.getSimpleName)

def getThriftSocialProof(

tweetSocialProof: SocialProofJavaResult

): Map[SocialProofThriftType, Seq[Long]] = {

Option(tweetSocialProof.getSocialProof) match {

case Some(socialProof) if socialProof.isEmpty =>

stats.counter(Stats.EmptyResult).incr()

Map.empty[SocialProofThriftType, Seq[Long]]

case Some(socialProof) if !socialProof.isEmpty =>

socialProof.asScala.map {

case (socialProofType, connectingUsers) =>

(

SocialProofThriftType(socialProofType.toInt),

connectingUsers.asScala.map { Long2long }.toSeq)

}.toMap

case \_ =>

throw new Exception("TweetSocialProofHandler gets wrong TweetSocialProof response")

}

}

def apply(request: SocialProofThriftRequest): Future[SocialProofThriftResponse] = {

StatsUtil.trackBlockStats(stats) {

if (decider.tweetSocialProof) {

val socialProofsFuture = tweetSocialProofRunner(request)

socialProofsFuture map { socialProofs: Seq[RecommendationInfo] =>

stats.counter(Stats.Served).incr(socialProofs.size)

SocialProofThriftResponse(

socialProofs.flatMap { tweetSocialProof: RecommendationInfo =>

val tweetSocialProofJavaResult = tweetSocialProof.asInstanceOf[SocialProofJavaResult]

Some(

TweetRecommendation(

tweetSocialProofJavaResult.getNode,

tweetSocialProofJavaResult.getWeight,

getThriftSocialProof(tweetSocialProofJavaResult)

)

)

}

)

}

} else {

Future.value(SocialProofThriftResponse())

}

}

}

}