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.RecommendationType

import com.twitter.graphjet.algorithms.counting.tweet.TweetMetadataRecommendationInfo

import com.twitter.graphjet.algorithms.counting.tweet.TweetRecommendationInfo

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

import com.twitter.recos.util.Stats

import com.twitter.servo.request.\_

import com.twitter.util.Future

/\*\*

\* Implementation of the Thrift-defined service interface.

\*

\* A wrapper of magicRecsRunner.

\*/

class RecommendationHandler(

tweetRecsRunner: TweetRecommendationsRunner,

statsReceiver: StatsReceiver)

extends RequestHandler[RecommendTweetEntityRequest, RecommendTweetEntityResponse] {

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

private val socialProofHydrator = new SocialProofHydrator(stats)

override def apply(request: RecommendTweetEntityRequest): Future[RecommendTweetEntityResponse] = {

val scopedStats: StatsReceiver = stats.scope(request.displayLocation.toString)

StatsUtil.trackBlockStats(scopedStats) {

val candidatesFuture = tweetRecsRunner.apply(request)

candidatesFuture.map { candidates =>

if (candidates.isEmpty) scopedStats.counter(Stats.EmptyResult).incr()

else scopedStats.counter(Stats.Served).incr(candidates.size)

RecommendTweetEntityResponse(candidates.flatMap {

\_ match {

case tweetRec: TweetRecommendationInfo =>

Some(

UserTweetEntityRecommendationUnion.TweetRec(

TweetRecommendation(

tweetRec.getRecommendation,

tweetRec.getWeight,

socialProofHydrator.addTweetSocialProofByType(tweetRec),

socialProofHydrator.addTweetSocialProofs(tweetRec)

)

)

)

case tweetMetadataRec: TweetMetadataRecommendationInfo =>

if (tweetMetadataRec.getRecommendationType == RecommendationType.HASHTAG) {

Some(

UserTweetEntityRecommendationUnion.HashtagRec(

HashtagRecommendation(

tweetMetadataRec.getRecommendation,

tweetMetadataRec.getWeight,

socialProofHydrator.addMetadataSocialProofByType(tweetMetadataRec)

)

)

)

} else if (tweetMetadataRec.getRecommendationType == RecommendationType.URL) {

Some(

UserTweetEntityRecommendationUnion.UrlRec(

UrlRecommendation(

tweetMetadataRec.getRecommendation,

tweetMetadataRec.getWeight,

socialProofHydrator.addMetadataSocialProofByType(tweetMetadataRec)

)

)

)

} else {

None: Option[UserTweetEntityRecommendationUnion]

}

case \_ => None

}

})

}

}

}

}