package com.twitter.recos.graph\_common

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.recos.recos\_common.thriftscala.{

SocialProofType,

GetRecentEdgesRequest,

GetRecentEdgesResponse,

NodeInfo,

RecentEdge

}

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

import com.twitter.servo.request.\_

import com.twitter.util.Future

/\*\*

\* Implementation of the Thrift-defined service interface.

\*/

class LeftNodeEdgesHandler(graphHelper: BipartiteGraphHelper, statsReceiver: StatsReceiver)

extends RequestHandler[GetRecentEdgesRequest, GetRecentEdgesResponse] {

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

private val CLICK = 0

private val FAVORITE = 1

private val RETWEET = 2

private val REPLY = 3

private val TWEET = 4

override def apply(request: GetRecentEdgesRequest): Future[GetRecentEdgesResponse] = {

trackFutureBlockStats(stats) {

val recentEdges = graphHelper.getLeftNodeEdges(request.requestId).flatMap {

case (node, engagementType) if engagementType == CLICK =>

Some(RecentEdge(node, SocialProofType.Click))

case (node, engagementType) if engagementType == FAVORITE =>

Some(RecentEdge(node, SocialProofType.Favorite))

case (node, engagementType) if engagementType == RETWEET =>

Some(RecentEdge(node, SocialProofType.Retweet))

case (node, engagementType) if engagementType == REPLY =>

Some(RecentEdge(node, SocialProofType.Reply))

case (node, engagementType) if engagementType == TWEET =>

Some(RecentEdge(node, SocialProofType.Tweet))

case \_ =>

None

}

Future.value(GetRecentEdgesResponse(recentEdges))

}

}

}

class RightNodeInfoHandler(graphHelper: BipartiteGraphHelper, statsReceiver: StatsReceiver)

extends RequestHandler[Long, NodeInfo] {

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

override def apply(rightNode: Long): Future[NodeInfo] = {

trackFutureBlockStats(stats) {

val edges = graphHelper.getRightNodeEdges(rightNode)

Future.value(NodeInfo(edges = edges))

}

}

}