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

import com.twitter.finagle.thrift.ClientId

import com.twitter.finagle.tracing.Trace

import com.twitter.finagle.tracing.TraceId

import com.twitter.recos.decider.EndpointLoadShedder

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

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

import com.twitter.util.Duration

import com.twitter.util.Future

import com.twitter.util.Timer

import scala.concurrent.duration.MILLISECONDS

import com.twitter.logging.Logger

import com.twitter.recos.user\_tweet\_graph.relatedTweetHandlers.TweetBasedRelatedTweetsHandler

import com.twitter.recos.user\_tweet\_graph.relatedTweetHandlers.ProducerBasedRelatedTweetsHandler

import com.twitter.recos.user\_tweet\_graph.relatedTweetHandlers.ConsumersBasedRelatedTweetsHandler

import com.twitter.simclusters\_v2.common.TweetId

import com.twitter.simclusters\_v2.common.UserId

object UserTweetGraph {

def traceId: TraceId = Trace.id

def clientId: Option[ClientId] = ClientId.current

}

class UserTweetGraph(

tweetBasedRelatedTweetsHandler: TweetBasedRelatedTweetsHandler,

producerBasedRelatedTweetsHandler: ProducerBasedRelatedTweetsHandler,

consumersBasedRelatedTweetsHandler: ConsumersBasedRelatedTweetsHandler,

endpointLoadShedder: EndpointLoadShedder

)(

implicit timer: Timer)

extends thriftscala.UserTweetGraph.MethodPerEndpoint {

private val defaultTimeout: Duration = Duration(50, MILLISECONDS)

private val EmptyResponse = Future.value(RelatedTweetResponse())

private val EmptyFeatureResponse = Future.value(UserTweetFeatureResponse())

private val log = Logger()

override def recommendTweets(request: RecommendTweetRequest): Future[RecommendTweetResponse] =

Future.value(RecommendTweetResponse())

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

Future.value(GetRecentEdgesResponse())

override def getRightNode(tweet: Long): Future[NodeInfo] = Future.value(NodeInfo())

// deprecated

override def relatedTweets(request: RelatedTweetRequest): Future[RelatedTweetResponse] =

EmptyResponse

override def tweetBasedRelatedTweets(

request: TweetBasedRelatedTweetRequest

): Future[RelatedTweetResponse] =

endpointLoadShedder("tweetBasedRelatedTweets") {

tweetBasedRelatedTweetsHandler(request).raiseWithin(defaultTimeout)

}.rescue {

case EndpointLoadShedder.LoadSheddingException =>

EmptyResponse

case e =>

log.info("user-tweet-graph\_tweetBasedRelatedTweets" + e)

EmptyResponse

}

override def producerBasedRelatedTweets(

request: ProducerBasedRelatedTweetRequest

): Future[RelatedTweetResponse] =

endpointLoadShedder("producerBasedRelatedTweets") {

producerBasedRelatedTweetsHandler(request).raiseWithin(defaultTimeout)

}.rescue {

case EndpointLoadShedder.LoadSheddingException =>

EmptyResponse

case e =>

log.info("user-tweet-graph\_producerBasedRelatedTweets" + e)

EmptyResponse

}

override def consumersBasedRelatedTweets(

request: ConsumersBasedRelatedTweetRequest

): Future[RelatedTweetResponse] =

endpointLoadShedder("consumersBasedRelatedTweets") {

consumersBasedRelatedTweetsHandler(request).raiseWithin(defaultTimeout)

}.rescue {

case EndpointLoadShedder.LoadSheddingException =>

EmptyResponse

case e =>

log.info("user-tweet-graph\_consumersBasedRelatedTweets" + e)

EmptyResponse

}

// deprecated

override def userTweetFeatures(

userId: UserId,

tweetId: TweetId

): Future[UserTweetFeatureResponse] =

EmptyFeatureResponse

}