package com.twitter.home\_mixer.product.scored\_tweets.feature\_hydrator

import com.google.inject.name.Named

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.home\_mixer.param.HomeMixerInjectionNames.RealTimeInteractionGraphUserVertexCache

import com.twitter.home\_mixer.util.ObservedKeyValueResultHandler

import com.twitter.product\_mixer.core.feature.Feature

import com.twitter.product\_mixer.core.feature.featuremap.FeatureMap

import com.twitter.product\_mixer.core.feature.featuremap.FeatureMapBuilder

import com.twitter.product\_mixer.core.functional\_component.feature\_hydrator.QueryFeatureHydrator

import com.twitter.product\_mixer.core.model.common.identifier.FeatureHydratorIdentifier

import com.twitter.product\_mixer.core.pipeline.PipelineQuery

import com.twitter.servo.cache.ReadCache

import com.twitter.stitch.Stitch

import com.twitter.wtf.real\_time\_interaction\_graph.{thriftscala => ig}

import javax.inject.Inject

import javax.inject.Singleton

object RealTimeInteractionGraphUserVertexQueryFeature

extends Feature[PipelineQuery, Option[ig.UserVertex]]

@Singleton

class RealTimeInteractionGraphUserVertexQueryFeatureHydrator @Inject() (

@Named(RealTimeInteractionGraphUserVertexCache) client: ReadCache[Long, ig.UserVertex],

override val statsReceiver: StatsReceiver)

extends QueryFeatureHydrator[PipelineQuery]

with ObservedKeyValueResultHandler {

override val identifier: FeatureHydratorIdentifier =

FeatureHydratorIdentifier("RealTimeInteractionGraphUserVertex")

override val features: Set[Feature[\_, \_]] = Set(RealTimeInteractionGraphUserVertexQueryFeature)

override val statScope: String = identifier.toString

override def hydrate(query: PipelineQuery): Stitch[FeatureMap] = {

val userId = query.getRequiredUserId

Stitch.callFuture(

client.get(Seq(userId)).map { results =>

val feature = observedGet(key = Some(userId), keyValueResult = results)

FeatureMapBuilder()

.add(RealTimeInteractionGraphUserVertexQueryFeature, feature)

.build()

}

)

}

}