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

import com.twitter.finagle.stats.StatsReceiver

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

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.search.common.constants.{thriftscala => scc}

import com.twitter.servo.repository.KeyValueRepository

import com.twitter.stitch.Stitch

import javax.inject.Inject

import javax.inject.Named

import javax.inject.Singleton

object UserLanguagesFeature extends Feature[PipelineQuery, Seq[scc.ThriftLanguage]]

@Singleton

case class UserLanguagesFeatureHydrator @Inject() (

@Named(UserLanguagesRepository) client: KeyValueRepository[Seq[Long], Long, Seq[

scc.ThriftLanguage

]],

statsReceiver: StatsReceiver)

extends QueryFeatureHydrator[PipelineQuery]

with ObservedKeyValueResultHandler {

override val identifier: FeatureHydratorIdentifier = FeatureHydratorIdentifier("UserLanguages")

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

override val statScope: String = identifier.toString

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

val key = query.getRequiredUserId

Stitch.callFuture(client(Seq(key))).map { result =>

val feature =

observedGet(key = Some(key), keyValueResult = result).map(\_.getOrElse(Seq.empty))

FeatureMapBuilder()

.add(UserLanguagesFeature, feature)

.build()

}

}

}