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

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

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

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.socialgraph.{thriftscala => sg}

import com.twitter.stitch.Stitch

import com.twitter.stitch.socialgraph.SocialGraph

import javax.inject.Inject

import javax.inject.Singleton

case object ListIdsFeature extends FeatureWithDefaultOnFailure[PipelineQuery, Seq[Long]] {

override val defaultValue: Seq[Long] = Seq.empty

}

@Singleton

class ListIdsQueryFeatureHydrator @Inject() (socialGraph: SocialGraph)

extends QueryFeatureHydrator[PipelineQuery] {

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

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

private val MaxListsToFetch = 20

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

val userId = query.getRequiredUserId

val ownedSubscribedRequest = sg.IdsRequest(

relationships = Seq(

sg.SrcRelationship(userId, sg.RelationshipType.ListIsSubscriber, hasRelationship = true),

sg.SrcRelationship(userId, sg.RelationshipType.ListOwning, hasRelationship = true)

),

pageRequest = Some(sg.PageRequest(selectAll = Some(false), count = Some(MaxListsToFetch))),

context = Some(

sg.LookupContext(

includeInactive = false,

performUnion = Some(true),

includeAll = Some(false)

)

)

)

socialGraph.ids(ownedSubscribedRequest).map { response =>

FeatureMapBuilder().add(ListIdsFeature, response.ids).build()

}

}

}