package com.twitter.home\_mixer.product.list\_recommended\_users.feature\_hydrator

import com.twitter.home\_mixer.model.request.HasListId

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

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

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

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

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 RecentListMembersFeature extends FeatureWithDefaultOnFailure[PipelineQuery, Seq[Long]] {

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

}

@Singleton

class RecentListMembersQueryFeatureHydrator @Inject() (socialGraph: SocialGraph)

extends QueryFeatureHydrator[PipelineQuery with HasListId] {

override val identifier: FeatureHydratorIdentifier =

FeatureHydratorIdentifier("RecentListMembers")

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

private val MaxRecentMembers = 10

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

val request = sg.IdsRequest(

relationships = Seq(sg

.SrcRelationship(query.listId, sg.RelationshipType.ListHasMember, hasRelationship = true)),

pageRequest = Some(sg.PageRequest(selectAll = Some(true), count = Some(MaxRecentMembers)))

)

socialGraph.ids(request).map(\_.ids).map { listMembers =>

FeatureMapBuilder().add(RecentListMembersFeature, listMembers).build()

}

}

}