package com.twitter.follow\_recommendations.common.candidate\_sources.real\_graph

import com.twitter.follow\_recommendations.common.clients.real\_time\_real\_graph.RealTimeRealGraphClient

import com.twitter.follow\_recommendations.common.models.CandidateUser

import com.twitter.hermit.model.Algorithm

import com.twitter.product\_mixer.core.functional\_component.candidate\_source.CandidateSource

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

import com.twitter.product\_mixer.core.model.marshalling.request.HasClientContext

import com.twitter.stitch.Stitch

import com.twitter.timelines.configapi.HasParams

import javax.inject.Inject

import javax.inject.Singleton

/\*\*

\* This source gets the already followed edges from the real graph column as a candidate source.

\*/

@Singleton

class RealGraphSource @Inject() (

realGraph: RealTimeRealGraphClient)

extends CandidateSource[HasParams with HasClientContext, CandidateUser] {

override val identifier: CandidateSourceIdentifier = RealGraphSource.Identifier

override def apply(request: HasParams with HasClientContext): Stitch[Seq[CandidateUser]] = {

request.getOptionalUserId

.map { userId =>

realGraph.getRealGraphWeights(userId).map { scoreMap =>

scoreMap.map {

case (candidateId, realGraphScore) =>

CandidateUser(id = candidateId, score = Some(realGraphScore))

.withCandidateSource(identifier)

}.toSeq

}

}.getOrElse(Stitch.Nil)

}

}

object RealGraphSource {

val Identifier: CandidateSourceIdentifier = CandidateSourceIdentifier(

Algorithm.RealGraphFollowed.toString)

}