package com.twitter.home\_mixer.product.list\_recommended\_users.candidate\_source

import com.twitter.hermit.candidate.{thriftscala => t}

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

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

import com.twitter.stitch.Stitch

import com.twitter.strato.client.Fetcher

import com.twitter.strato.generated.client.recommendations.similarity.SimilarUsersBySimsOnUserClientColumn

import javax.inject.Inject

import javax.inject.Singleton

@Singleton

class SimilarityBasedUsersCandidateSource @Inject() (

similarUsersBySimsOnUserClientColumn: SimilarUsersBySimsOnUserClientColumn)

extends CandidateSource[Seq[Long], t.Candidate] {

override val identifier: CandidateSourceIdentifier =

CandidateSourceIdentifier("SimilarityBasedUsers")

private val fetcher: Fetcher[Long, Unit, t.Candidates] =

similarUsersBySimsOnUserClientColumn.fetcher

private val MaxCandidatesToKeep = 4000

override def apply(request: Seq[Long]): Stitch[Seq[t.Candidate]] = {

Stitch

.collect {

request.map { userId =>

fetcher

.fetch(userId, Unit).map { result =>

result.v.map(\_.candidates).getOrElse(Seq.empty)

}.map { candidates =>

val sortedCandidates = candidates.sortBy(-\_.score)

sortedCandidates.take(MaxCandidatesToKeep)

}

}

}.map(\_.flatten)

}

}