package com.twitter.follow\_recommendations.common.candidate\_sources.sims

import com.twitter.escherbird.util.stitchcache.StitchCache

import com.twitter.finagle.stats.StatsReceiver

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

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

import com.twitter.hermit.candidate.thriftscala.Candidates

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.timelines.configapi.HasParams

import com.twitter.util.Duration

import java.lang.{Long => JLong}

class CacheBasedSimsStore(

id: CandidateSourceIdentifier,

fetcher: Fetcher[Long, Unit, Candidates],

maxCacheSize: Int,

cacheTtl: Duration,

statsReceiver: StatsReceiver)

extends CandidateSource[HasParams with HasSimilarToContext, CandidateUser] {

override val identifier: CandidateSourceIdentifier = id

private def getUsersFromSimsSource(userId: JLong): Stitch[Option[Candidates]] = {

fetcher

.fetch(userId)

.map(\_.v)

}

private val simsCache = StitchCache[JLong, Option[Candidates]](

maxCacheSize = maxCacheSize,

ttl = cacheTtl,

statsReceiver = statsReceiver,

underlyingCall = getUsersFromSimsSource

)

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

Stitch

.traverse(request.similarToUserIds) { userId =>

simsCache.readThrough(userId).map { candidatesOpt =>

candidatesOpt

.map { candidates =>

StratoBasedSimsCandidateSource.map(userId, candidates)

}.getOrElse(Nil)

}

}.map(\_.flatten.distinct.map(\_.withCandidateSource(identifier)))

}

}