package com.twitter.product\_mixer.component\_library.candidate\_source.social\_graph

import com.twitter.product\_mixer.component\_library.model.candidate.CursorType

import com.twitter.product\_mixer.component\_library.model.candidate.NextCursor

import com.twitter.product\_mixer.component\_library.model.candidate.PreviousCursor

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

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

import com.twitter.socialgraph.thriftscala

import com.twitter.socialgraph.thriftscala.IdsRequest

import com.twitter.socialgraph.thriftscala.IdsResult

import com.twitter.socialgraph.util.ByteBufferUtil

import com.twitter.strato.client.Fetcher

import javax.inject.Inject

import javax.inject.Singleton

sealed trait SocialgraphResponse

case class SocialgraphResult(id: Long) extends SocialgraphResponse

case class SocialgraphCursor(cursor: Long, cursorType: CursorType) extends SocialgraphResponse

@Singleton

class SocialgraphCandidateSource @Inject() (

override val fetcher: Fetcher[thriftscala.IdsRequest, Option[

thriftscala.RequestContext

], thriftscala.IdsResult])

extends StratoKeyViewFetcherSource[

thriftscala.IdsRequest,

Option[thriftscala.RequestContext],

thriftscala.IdsResult,

SocialgraphResponse

] {

override val identifier: CandidateSourceIdentifier = CandidateSourceIdentifier("Socialgraph")

override def stratoResultTransformer(

stratoKey: IdsRequest,

stratoResult: IdsResult

): Seq[SocialgraphResponse] = {

val prevCursor =

SocialgraphCursor(ByteBufferUtil.toLong(stratoResult.pageResult.prevCursor), PreviousCursor)

/\* When an end cursor is passed to Socialgraph,

\* Socialgraph returns the start cursor. To prevent

\* clients from circularly fetching the timeline again,

\* if we see a start cursor returned from Socialgraph,

\* we replace it with an end cursor.

\*/

val nextCursor = ByteBufferUtil.toLong(stratoResult.pageResult.nextCursor) match {

case SocialgraphCursorConstants.StartCursor =>

SocialgraphCursor(SocialgraphCursorConstants.EndCursor, NextCursor)

case cursor => SocialgraphCursor(cursor, NextCursor)

}

stratoResult.ids

.map { id =>

SocialgraphResult(id)

} ++ Seq(nextCursor, prevCursor)

}

}