package com.twitter.usersignalservice.signals

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.socialgraph.thriftscala.RelationshipType

import com.twitter.socialgraph.thriftscala.SocialGraphService

import com.twitter.twistly.common.UserId

import com.twitter.usersignalservice.base.BaseSignalFetcher

import com.twitter.usersignalservice.base.Query

import com.twitter.usersignalservice.signals.common.SGSUtils

import com.twitter.usersignalservice.thriftscala.Signal

import com.twitter.usersignalservice.thriftscala.SignalType

import com.twitter.util.Future

import com.twitter.util.Timer

import javax.inject.Inject

import javax.inject.Singleton

@Singleton

case class AccountFollowsFetcher @Inject() (

sgsClient: SocialGraphService.MethodPerEndpoint,

timer: Timer,

stats: StatsReceiver)

extends BaseSignalFetcher {

override type RawSignalType = Signal

override val name: String = this.getClass.getCanonicalName

override val statsReceiver: StatsReceiver = stats.scope(this.name)

override def getRawSignals(

userId: UserId

): Future[Option[Seq[RawSignalType]]] = {

SGSUtils.getSGSRawSignals(

userId,

sgsClient,

RelationshipType.Following,

SignalType.AccountFollow)

}

override def process(

query: Query,

rawSignals: Future[Option[Seq[RawSignalType]]]

): Future[Option[Seq[Signal]]] = {

rawSignals.map(\_.map(\_.take(query.maxResults.getOrElse(Int.MaxValue))))

}

}