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

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

import com.twitter.stitch.Stitch

import com.twitter.timelines.configapi.HasParams

import com.twitter.timelines.configapi.Param

import com.twitter.finagle.stats.StatsReceiver

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

/\*\*

\* A wrapper of CandidateSource to make it easier to do experimentation

\* on new candidate generation algorithms

\*

\* @param baseSource base candidate source

\* @param darkreadAlgorithmParam controls whether or not to darkread candidates (fetch them even if they will not be included)

\* @param keepCandidatesParam controls whether or not to keep candidates from the base source

\* @param resultCountThresholdParam controls how many results the source must return to bucket the user and return results (greater-than-or-equal-to)

\* @tparam T request type. it must extend HasParams

\* @tparam V value type

\*/

class ExperimentalCandidateSource[T <: HasParams, V](

baseSource: CandidateSource[T, V],

darkreadAlgorithmParam: Param[Boolean],

keepCandidatesParam: Param[Boolean],

resultCountThresholdParam: Param[Int],

baseStatsReceiver: StatsReceiver)

extends CandidateSource[T, V] {

override val identifier: CandidateSourceIdentifier = baseSource.identifier

private[base] val statsReceiver =

baseStatsReceiver.scope(s"Experimental/${identifier.name}")

private[base] val requestsCounter = statsReceiver.counter("requests")

private[base] val resultCountGreaterThanThresholdCounter =

statsReceiver.counter("with\_results\_at\_or\_above\_count\_threshold")

private[base] val keepResultsCounter = statsReceiver.counter("keep\_results")

private[base] val discardResultsCounter = statsReceiver.counter("discard\_results")

override def apply(request: T): Stitch[Seq[V]] = {

if (request.params(darkreadAlgorithmParam)) {

requestsCounter.incr()

fetchFromCandidateSourceAndProcessResults(request)

} else {

Stitch.Nil

}

}

private def fetchFromCandidateSourceAndProcessResults(request: T): Stitch[Seq[V]] = {

baseSource(request).map { results =>

if (results.length >= request.params(resultCountThresholdParam)) {

processResults(results, request.params(keepCandidatesParam))

} else {

Nil

}

}

}

private def processResults(results: Seq[V], keepResults: Boolean): Seq[V] = {

resultCountGreaterThanThresholdCounter.incr()

if (keepResults) {

keepResultsCounter.incr()

results

} else {

discardResultsCounter.incr()

Nil

}

}

}