package com.twitter.frigate.pushservice.refresh\_handler.cross

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.frigate.common.base.CandidateDetails

import com.twitter.frigate.common.util.MRNtabCopy

import com.twitter.frigate.common.util.MRPushCopy

import com.twitter.frigate.pushservice.model.PushTypes.RawCandidate

import com.twitter.frigate.pushservice.model.candidate.CopyIds

import com.twitter.util.Future

/\*\*

\* @param statsReceiver - stats receiver object

\*/

class CandidateCopyExpansion(statsReceiver: StatsReceiver)

extends BaseCopyFramework(statsReceiver) {

/\*\*

\*

\* Given a [[CandidateDetails]] object representing a push recommendation candidate this method

\* expands it to multiple candidates, each tagged with a push copy id and ntab copy id to

\* represent the eligible copies for the given recommendation candidate

\*

\* @param candidateDetails - [[CandidateDetails]] objects containing a recommendation candidate

\*

\* @return - list of tuples of [[PushTypes.RawCandidate]] and [[CopyIds]]

\*/

private final def crossCandidateDetailsWithCopyId(

candidateDetails: CandidateDetails[RawCandidate]

): Future[Seq[(CandidateDetails[RawCandidate], CopyIds)]] = {

val eligibleCopyPairs = getEligiblePushAndNtabCopiesFromCandidate(candidateDetails.candidate)

val copyPairs = eligibleCopyPairs.map(\_.map {

case (pushCopy: MRPushCopy, ntabCopy: Option[MRNtabCopy]) =>

CopyIds(

pushCopyId = Some(pushCopy.copyId),

ntabCopyId = ntabCopy.map(\_.copyId)

)

})

copyPairs.map(\_.map((candidateDetails, \_)))

}

/\*\*

\*

\* This method takes as input a list of [[CandidateDetails]] objects which contain the push

\* recommendation candidates for a given target user. It expands each input candidate into

\* multiple candidates, each tagged with a push copy id and ntab copy id to represent the eligible

\* copies for the given recommendation candidate

\*

\* @param candidateDetailsSeq - list of fetched candidates for push recommendation

\* @return - list of tuples of [[RawCandidate]] and [[CopyIds]]

\*/

final def expandCandidatesWithCopyId(

candidateDetailsSeq: Seq[CandidateDetails[RawCandidate]]

): Future[Seq[(CandidateDetails[RawCandidate], CopyIds)]] =

Future.collect(candidateDetailsSeq.map(crossCandidateDetailsWithCopyId)).map(\_.flatten)

}