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

import com.twitter.finagle.stats.StatsReceiver

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.util.Future

abstract class BaseCopyFramework(statsReceiver: StatsReceiver) {

private val NoAvailableCopyStat = statsReceiver.scope("no\_copy\_for\_crt")

private val NoAvailableNtabCopyStat = statsReceiver.scope("no\_ntab\_copy")

/\*\*

\* Instantiate push copy filters

\*/

protected final val copyFilters = new CopyFilters(statsReceiver.scope("filters"))

/\*\*

\*

\* The following method fetches all the push copies for a [[com.twitter.frigate.thriftscala.CommonRecommendationType]]

\* associated with a candidate and then filters the eligible copies based on

\* [[PushTypes.PushCandidate]] features. These filters are defined in

\* [[CopyFilters]]

\*

\* @param rawCandidate - [[RawCandidate]] object representing a recommendation candidate

\*

\* @return - set of eligible push copies for a given candidate

\*/

protected[cross] final def getEligiblePushCopiesFromCandidate(

rawCandidate: RawCandidate

): Future[Seq[MRPushCopy]] = {

val pushCopiesFromRectype = CandidateToCopy.getPushCopiesFromRectype(rawCandidate.commonRecType)

if (pushCopiesFromRectype.isEmpty) {

NoAvailableCopyStat.counter(rawCandidate.commonRecType.name).incr()

throw new IllegalStateException(s"No Copy defined for CRT: " + rawCandidate.commonRecType)

}

pushCopiesFromRectype

.map(pushCopySet => copyFilters.execute(rawCandidate, pushCopySet.toSeq))

.getOrElse(Future.value(Seq.empty))

}

/\*\*

\*

\* This method essentially forms the base for cross-step for the MagicRecs Copy Framework. Given

\* a recommendation type this returns a set of tuples wherein each tuple is a pair of push and

\* ntab copy eligible for the said recommendation type

\*

\* @param rawCandidate - [[RawCandidate]] object representing a recommendation candidate

\* @return - Set of eligible [[MRPushCopy]], Option[[MRNtabCopy]] for a given recommendation type

\*/

protected[cross] final def getEligiblePushAndNtabCopiesFromCandidate(

rawCandidate: RawCandidate

): Future[Seq[(MRPushCopy, Option[MRNtabCopy])]] = {

val eligiblePushCopies = getEligiblePushCopiesFromCandidate(rawCandidate)

eligiblePushCopies.map { pushCopies =>

val setBuilder = Set.newBuilder[(MRPushCopy, Option[MRNtabCopy])]

pushCopies.foreach { pushCopy =>

val ntabCopies = CandidateToCopy.getNtabcopiesFromPushcopy(pushCopy)

val pushNtabCopyPairs = ntabCopies match {

case Some(ntabCopySet) =>

if (ntabCopySet.isEmpty) {

NoAvailableNtabCopyStat.counter(s"copy\_id: ${pushCopy.copyId}").incr()

Set(pushCopy -> None)

} // push copy only

else ntabCopySet.map(pushCopy -> Some(\_))

case None =>

Set.empty[(MRPushCopy, Option[MRNtabCopy])] // no push or ntab copy

}

setBuilder ++= pushNtabCopyPairs

}

setBuilder.result().toSeq

}

}

}