package com.twitter.frigate.pushservice.adaptor

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.frigate.common.base.\_

import com.twitter.frigate.common.candidate.\_

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

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

import com.twitter.frigate.pushservice.params.PushParams

import com.twitter.frigate.pushservice.util.PushDeviceUtil

import com.twitter.stitch.tweetypie.TweetyPie.TweetyPieResult

import com.twitter.storehaus.ReadableStore

import com.twitter.util.Future

object GenericCandidates {

type Target =

TargetUser

with UserDetails

with TargetDecider

with TargetABDecider

with TweetImpressionHistory

with HTLVisitHistory

with MaxTweetAge

with NewUserDetails

with FrigateHistory

with TargetWithSeedUsers

}

case class GenericCandidateAdaptor(

genericCandidates: CandidateSource[GenericCandidates.Target, Candidate],

tweetyPieStore: ReadableStore[Long, TweetyPieResult],

tweetyPieStoreNoVF: ReadableStore[Long, TweetyPieResult],

stats: StatsReceiver)

extends CandidateSource[Target, RawCandidate]

with CandidateSourceEligible[Target, RawCandidate] {

override val name: String = genericCandidates.name

private def generateTweetFavCandidate(

\_target: Target,

\_tweetId: Long,

\_socialContextActions: Seq[SocialContextAction],

socialContextActionsAllTypes: Seq[SocialContextAction],

\_tweetyPieResult: Option[TweetyPieResult]

): RawCandidate = {

new RawCandidate with TweetFavoriteCandidate {

override val socialContextActions = \_socialContextActions

override val socialContextAllTypeActions =

socialContextActionsAllTypes

val tweetId = \_tweetId

val target = \_target

val tweetyPieResult = \_tweetyPieResult

}

}

private def generateTweetRetweetCandidate(

\_target: Target,

\_tweetId: Long,

\_socialContextActions: Seq[SocialContextAction],

socialContextActionsAllTypes: Seq[SocialContextAction],

\_tweetyPieResult: Option[TweetyPieResult]

): RawCandidate = {

new RawCandidate with TweetRetweetCandidate {

override val socialContextActions = \_socialContextActions

override val socialContextAllTypeActions = socialContextActionsAllTypes

val tweetId = \_tweetId

val target = \_target

val tweetyPieResult = \_tweetyPieResult

}

}

override def get(inputTarget: Target): Future[Option[Seq[RawCandidate]]] = {

genericCandidates.get(inputTarget).map { candidatesOpt =>

candidatesOpt

.map { candidates =>

val candidatesSeq =

candidates.collect {

case tweetRetweet: TweetRetweetCandidate

if inputTarget.params(PushParams.MRTweetRetweetRecsParam) =>

generateTweetRetweetCandidate(

inputTarget,

tweetRetweet.tweetId,

tweetRetweet.socialContextActions,

tweetRetweet.socialContextAllTypeActions,

tweetRetweet.tweetyPieResult)

case tweetFavorite: TweetFavoriteCandidate

if inputTarget.params(PushParams.MRTweetFavRecsParam) =>

generateTweetFavCandidate(

inputTarget,

tweetFavorite.tweetId,

tweetFavorite.socialContextActions,

tweetFavorite.socialContextAllTypeActions,

tweetFavorite.tweetyPieResult)

}

candidatesSeq.foreach { candidate =>

stats.counter(s"${candidate.commonRecType}\_count").incr()

}

candidatesSeq

}

}

}

override def isCandidateSourceAvailable(target: Target): Future[Boolean] = {

PushDeviceUtil.isRecommendationsEligible(target).map { isAvailable =>

isAvailable && target.params(PushParams.GenericCandidateAdaptorDecider)

}

}

}