package com.twitter.cr\_mixer.filter

import com.twitter.cr\_mixer.model.CandidateGeneratorQuery

import com.twitter.cr\_mixer.model.InitialCandidate

import com.twitter.cr\_mixer.model.ModuleNames

import com.twitter.cr\_mixer.model.UtegTweetCandidateGeneratorQuery

import com.twitter.cr\_mixer.param.UtegTweetGlobalParams

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.frigate.common.util.StatsUtil

import com.twitter.simclusters\_v2.common.UserId

import com.twitter.storehaus.ReadableStore

import com.twitter.util.Future

import com.twitter.wtf.candidate.thriftscala.CandidateSeq

import javax.inject.Inject

import javax.inject.Named

import javax.inject.Singleton

/\*\*\*

\* Filters in-network tweets

\*/

@Singleton

case class InNetworkFilter @Inject() (

@Named(ModuleNames.RealGraphInStore) realGraphStoreMh: ReadableStore[UserId, CandidateSeq],

globalStats: StatsReceiver)

extends FilterBase {

override val name: String = this.getClass.getCanonicalName

import InNetworkFilter.\_

override type ConfigType = FilterConfig

private val stats: StatsReceiver = globalStats.scope(this.getClass.getCanonicalName)

private val filterCandidatesStats = stats.scope("filter\_candidates")

override def filter(

candidates: Seq[Seq[InitialCandidate]],

filterConfig: FilterConfig,

): Future[Seq[Seq[InitialCandidate]]] = {

StatsUtil.trackItemsStats(filterCandidatesStats) {

filterCandidates(candidates, filterConfig)

}

}

private def filterCandidates(

candidates: Seq[Seq[InitialCandidate]],

filterConfig: FilterConfig,

): Future[Seq[Seq[InitialCandidate]]] = {

if (!filterConfig.enableInNetworkFilter) {

Future.value(candidates)

} else {

filterConfig.userIdOpt match {

case Some(userId) =>

realGraphStoreMh

.get(userId).map(\_.map(\_.candidates.map(\_.userId)).getOrElse(Seq.empty).toSet).map {

realGraphInNetworkAuthorsSet =>

candidates.map(\_.filterNot { candidate =>

realGraphInNetworkAuthorsSet.contains(candidate.tweetInfo.authorId)

})

}

case None => Future.value(candidates)

}

}

}

override def requestToConfig[CGQueryType <: CandidateGeneratorQuery](

request: CGQueryType

): FilterConfig = {

request match {

case UtegTweetCandidateGeneratorQuery(userId, \_, \_, \_, \_, params, \_) =>

FilterConfig(Some(userId), params(UtegTweetGlobalParams.EnableInNetworkFilterParam))

case \_ => FilterConfig(None, false)

}

}

}

object InNetworkFilter {

case class FilterConfig(

userIdOpt: Option[UserId],

enableInNetworkFilter: Boolean)

}