package com.twitter.cr\_mixer.filter

import com.twitter.contentrecommender.thriftscala.TweetInfo

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

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

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

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

import com.twitter.util.Future

import javax.inject.Singleton

@Singleton

trait TweetInfoHealthFilterBase extends FilterBase {

override def name: String = this.getClass.getCanonicalName

override type ConfigType = HealthThreshold.Enum.Value

def thresholdToPropertyMap: Map[HealthThreshold.Enum.Value, TweetInfo => Option[Boolean]]

def getFilterParamFn: CandidateGeneratorQuery => HealthThreshold.Enum.Value

override def filter(

candidates: Seq[Seq[InitialCandidate]],

config: HealthThreshold.Enum.Value

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

Future.value(candidates.map { seq =>

seq.filter(p => thresholdToPropertyMap(config)(p.tweetInfo).getOrElse(true))

})

}

/\*\*

\* Build the config params here. passing in param() into the filter is strongly discouraged

\* because param() can be slow when called many times

\*/

override def requestToConfig[CGQueryType <: CandidateGeneratorQuery](

query: CGQueryType

): HealthThreshold.Enum.Value = {

query match {

case q: CrCandidateGeneratorQuery => getFilterParamFn(q)

case \_ => HealthThreshold.Enum.Off

}

}

}