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.param.GlobalParams

import com.twitter.snowflake.id.SnowflakeId

import com.twitter.util.Duration

import com.twitter.util.Future

import com.twitter.util.Time

import javax.inject.Singleton

import com.twitter.conversions.DurationOps.\_

@Singleton

case class TweetAgeFilter() extends FilterBase {

override val name: String = this.getClass.getCanonicalName

override type ConfigType = Duration

override def filter(

candidates: Seq[Seq[InitialCandidate]],

maxTweetAge: Duration

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

if (maxTweetAge >= 720.hours) {

Future.value(candidates)

} else {

// Tweet IDs are approximately chronological (see http://go/snowflake),

// so we are building the earliest tweet id once,

// and pass that as the value to filter candidates for each CandidateGenerationModel.

val earliestTweetId = SnowflakeId.firstIdFor(Time.now - maxTweetAge)

Future.value(candidates.map(\_.filter(\_.tweetId >= earliestTweetId)))

}

}

override def requestToConfig[CGQueryType <: CandidateGeneratorQuery](

query: CGQueryType

): Duration = {

query.params(GlobalParams.MaxTweetAgeHoursParam)

}

}