package com.twitter.product\_mixer.component\_library.filter

import com.twitter.product\_mixer.core.functional\_component.filter.Filter

import com.twitter.product\_mixer.core.functional\_component.filter.FilterResult

import com.twitter.product\_mixer.core.model.common.CandidateWithFeatures

import com.twitter.product\_mixer.core.model.common.UniversalNoun

import com.twitter.product\_mixer.core.model.common.identifier.FilterIdentifier

import com.twitter.product\_mixer.core.pipeline.PipelineQuery

import com.twitter.snowflake.id.SnowflakeId

import com.twitter.stitch.Stitch

import com.twitter.timelines.configapi.Param

import com.twitter.util.Duration

/\*\*

\* @param maxAgeParam Feature Switch configurable for convenience

\* @tparam Candidate The type of the candidates

\*/

case class SnowflakeIdAgeFilter[Candidate <: UniversalNoun[Long]](

maxAgeParam: Param[Duration])

extends Filter[PipelineQuery, Candidate] {

override val identifier: FilterIdentifier = FilterIdentifier("SnowflakeIdAge")

override def apply(

query: PipelineQuery,

candidates: Seq[CandidateWithFeatures[Candidate]]

): Stitch[FilterResult[Candidate]] = {

val maxAge = query.params(maxAgeParam)

val (keptCandidates, removedCandidates) = candidates

.map(\_.candidate)

.partition { filterCandidate =>

SnowflakeId.timeFromIdOpt(filterCandidate.id) match {

case Some(creationTime) =>

query.queryTime.since(creationTime) <= maxAge

case \_ => false

}

}

Stitch.value(FilterResult(kept = keptCandidates, removed = removedCandidates))

}

}