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

import com.twitter.product\_mixer.component\_library.model.cursor.UrtUnorderedExcludeIdsCursor

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.HasPipelineCursor

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

import com.twitter.stitch.Stitch

case class UrtUnorderedExcludeIdsCursorFilter[

Candidate <: UniversalNoun[Long],

Query <: PipelineQuery with HasPipelineCursor[UrtUnorderedExcludeIdsCursor]

]() extends Filter[Query, Candidate] {

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

override def apply(

query: Query,

candidates: Seq[CandidateWithFeatures[Candidate]]

): Stitch[FilterResult[Candidate]] = {

val excludeIds = query.pipelineCursor.map(\_.excludedIds.toSet).getOrElse(Set.empty)

val (kept, removed) =

candidates.map(\_.candidate).partition(candidate => !excludeIds.contains(candidate.id))

val filterResult = FilterResult(kept = kept, removed = removed)

Stitch.value(filterResult)

}

}