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.model.marshalling.request.HasExcludedIds

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

import com.twitter.stitch.Stitch

case class ExcludedIdsFilter[

Query <: PipelineQuery with HasExcludedIds,

Candidate <: UniversalNoun[Long]

]() extends Filter[Query, Candidate] {

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

override def apply(

query: Query,

candidates: Seq[CandidateWithFeatures[Candidate]]

): Stitch[FilterResult[Candidate]] = {

val (kept, removed) =

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

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

Stitch.value(filterResult)

}

}