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.stitch.Stitch

/\*\*

\* Predicate which will be applied to each candidate. True indicates that the candidate will be

\* @tparam Candidate - the type of the candidate

\*/

trait ShouldKeepCandidate[Candidate] {

def apply(candidate: Candidate): Boolean

}

object PredicateFilter {

/\*\*

\* Builds a simple Filter out of a predicate function from the candidate to a boolean. For clarity,

\* we recommend including the name of the shouldKeepCandidate parameter.

\*

\* {{{

\* Filter.fromPredicate(

\* FilterIdentifier("SomeFilter"),

\* shouldKeepCandidate = { candidate: UserCandidate => candidate.id % 2 == 0L }

\* )

\* }}}

\*

\* @param identifier A FilterIdentifier for the new filter

\* @param shouldKeepCandidate A predicate function from the candidate. Candidates will be kept

\* when this function returns True.

\*/

def fromPredicate[Candidate <: UniversalNoun[Any]](

identifier: FilterIdentifier,

shouldKeepCandidate: ShouldKeepCandidate[Candidate]

): Filter[PipelineQuery, Candidate] = {

val i = identifier

new Filter[PipelineQuery, Candidate] {

override val identifier: FilterIdentifier = i

/\*\*

\* Filter the list of candidates

\*

\* @return a FilterResult including both the list of kept candidate and the list of removed candidates

\*/

override def apply(

query: PipelineQuery,

candidates: Seq[CandidateWithFeatures[Candidate]]

): Stitch[FilterResult[Candidate]] = {

val (keptCandidates, removedCandidates) = candidates.map(\_.candidate).partition {

filterCandidate =>

shouldKeepCandidate(filterCandidate)

}

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

}

}

}

}