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

import com.twitter.product\_mixer.core.feature.Feature

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

object FeatureFilter {

/\*\*

\* Builds a Filter using the Feature name as the FilterIdentifier

\*

\* @see [[FeatureFilter.fromFeature(identifier, feature)]]

\*/

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

feature: Feature[Candidate, Boolean]

): Filter[PipelineQuery, Candidate] =

FeatureFilter.fromFeature(FilterIdentifier(feature.toString), feature)

/\*\*

\* Builds a Filter that keeps candidates when the provided Boolean Feature is present and True.

\* If the Feature is missing or False, the candidate is removed.

\*

\* {{{

\* Filter.fromFeature(

\* FilterIdentifier("SomeFilter"),

\* feature = SomeFeature

\* )

\* }}}

\*

\* @param identifier A FilterIdentifier for the new filter

\* @param feature A feature of [Candidate, Boolean] type used to determine whether Candidates will be kept

\* when this feature is present and true otherwise they will be removed.

\*/

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

identifier: FilterIdentifier,

feature: Feature[Candidate, Boolean]

): Filter[PipelineQuery, Candidate] = {

val i = identifier

new Filter[PipelineQuery, Candidate] {

override val identifier: FilterIdentifier = i

override def apply(

query: PipelineQuery,

candidates: Seq[CandidateWithFeatures[Candidate]]

): Stitch[FilterResult[Candidate]] = {

val (keptCandidates, removedCandidates) = candidates.partition { filterCandidate =>

filterCandidate.features.getOrElse(feature, false)

}

Stitch.value(

FilterResult(

kept = keptCandidates.map(\_.candidate),

removed = removedCandidates.map(\_.candidate)))

}

}

}

}