package com.twitter.product\_mixer.core.functional\_component.filter

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

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

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

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

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

/\*\*

\* Takes a sequence of candidates and can filter some out

\*

\* @note if you want to conditionally run a [[Filter]] you can use the mixin [[Filter.Conditionally]]

\* or to gate on a [[com.twitter.timelines.configapi.Param]] you can use [[com.twitter.product\_mixer.component\_library.filter.ParamGatedFilter]]

\*

\* @tparam Query The domain model for the query or request

\* @tparam Candidate The type of the candidates

\*/

trait Filter[-Query <: PipelineQuery, Candidate <: UniversalNoun[Any]]

extends Component

with SupportsConditionally[Query, Candidate] {

/\*\* @see [[FilterIdentifier]] \*/

override val identifier: FilterIdentifier

/\*\*

\* Filter the list of candidates

\*

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

\*/

def apply(

query: Query,

candidates: Seq[CandidateWithFeatures[Candidate]]

): Stitch[FilterResult[Candidate]]

}

object Filter {

/\*\*

\* Mixin for when you want to conditionally run a [[Filter]]

\*

\* This is a thin wrapper around [[common.Conditionally]] exposing a nicer API for the [[Filter]] specific use-case.

\*/

trait Conditionally[-Query <: PipelineQuery, Candidate <: UniversalNoun[Any]]

extends common.Conditionally[Input[Query, Candidate]] { \_: Filter[Query, Candidate] =>

/\*\* @see [[common.Conditionally.onlyIf]] \*/

def onlyIf(

query: Query,

candidates: Seq[CandidateWithFeatures[Candidate]]

): Boolean

override final def onlyIf(input: Input[Query, Candidate]): Boolean =

onlyIf(input.query, input.candidates)

}

/\*\* Type alias to obscure [[Filter.Input]] from customers \*/

type SupportsConditionally[-Query <: PipelineQuery, Candidate <: UniversalNoun[Any]] =

common.SupportsConditionally[Input[Query, Candidate]]

/\*\* A case class representing the input arguments to a [[Filter]], mostly for internal use \*/

case class Input[+Query <: PipelineQuery, +Candidate <: UniversalNoun[Any]](

query: Query,

candidates: Seq[CandidateWithFeatures[Candidate]])

}