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

import com.twitter.product\_mixer.core.functional\_component.common.CandidateScope

import com.twitter.product\_mixer.core.functional\_component.common.SpecificPipeline

import com.twitter.product\_mixer.core.functional\_component.common.SpecificPipelines

import com.twitter.product\_mixer.core.functional\_component.selector.Selector

import com.twitter.product\_mixer.core.functional\_component.selector.SelectorResult

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

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

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

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

object DropFilteredModuleItemCandidates {

def apply(candidatePipeline: CandidatePipelineIdentifier, filter: ShouldKeepCandidate) =

new DropFilteredModuleItemCandidates(SpecificPipeline(candidatePipeline), filter)

def apply(candidatePipelines: Set[CandidatePipelineIdentifier], filter: ShouldKeepCandidate) =

new DropFilteredModuleItemCandidates(SpecificPipelines(candidatePipelines), filter)

}

/\*\*

\* Limit candidates in modules from certain candidates sources to those which satisfy

\* the provided predicate.

\*

\* This acts like a [[DropFilteredCandidates]] but for modules in `remainingCandidates`

\* from any of the provided [[candidatePipelines]].

\*

\* @note this updates the module in the `remainingCandidates`

\*/

case class DropFilteredModuleItemCandidates(

override val pipelineScope: CandidateScope,

filter: ShouldKeepCandidate)

extends Selector[PipelineQuery] {

override def apply(

query: PipelineQuery,

remainingCandidates: Seq[CandidateWithDetails],

result: Seq[CandidateWithDetails]

): SelectorResult = {

val candidatesUpdated = remainingCandidates.map {

case module: ModuleCandidateWithDetails if pipelineScope.contains(module) =>

// this applies to all candidates in a module, even if they are from a different

// candidate source, which can happen if items are added to a module during selection

module.copy(candidates = module.candidates.filter(filter.apply))

case candidate => candidate

}

SelectorResult(remainingCandidates = candidatesUpdated, result = result)

}

}