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.selector.Selector

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

import CandidateScope.PartitionedCandidates

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.model.common.identifier.CandidatePipelineIdentifier

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

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

object InsertAppendResults {

def apply(candidatePipeline: CandidatePipelineIdentifier): InsertAppendResults[PipelineQuery] =

new InsertAppendResults(SpecificPipeline(candidatePipeline))

def apply(

candidatePipelines: Set[CandidatePipelineIdentifier]

): InsertAppendResults[PipelineQuery] = new InsertAppendResults(

SpecificPipelines(candidatePipelines))

}

/\*\*

\* Select all candidates from candidate pipeline(s) and append to the end of the result.

\*

\* @note that if multiple candidate pipelines are specified, their candidates will be added

\* to the result in the order in which they appear in the candidate pool. This ordering often

\* reflects the order in which the candidate pipelines were listed in the mixer/recommendations

\* pipeline, unless for example an UpdateSortCandidates selector was run prior to running

\* this selector which could change this ordering.

\*

\* @note if inserting results from multiple candidate pipelines (see note above related to ordering),

\* it is more performant to include all (or a subset) of the candidate pipelines in a single

\* InsertAppendResults, as opposed to calling InsertAppendResults individually for each

\* candidate pipeline because each selector does an O(n) pass on the candidate pool.

\*/

case class InsertAppendResults[-Query <: PipelineQuery](

override val pipelineScope: CandidateScope)

extends Selector[Query] {

override def apply(

query: Query,

remainingCandidates: Seq[CandidateWithDetails],

result: Seq[CandidateWithDetails]

): SelectorResult = {

val PartitionedCandidates(selectedCandidates, otherCandidates) =

pipelineScope.partition(remainingCandidates)

val resultUpdated = result ++ selectedCandidates

SelectorResult(remainingCandidates = otherCandidates, result = resultUpdated)

}

}