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.\_

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 InsertDynamicPositionResults {

def apply[Query <: PipelineQuery](

candidatePipeline: CandidatePipelineIdentifier,

dynamicInsertionPosition: DynamicInsertionPosition[Query],

): InsertDynamicPositionResults[Query] =

new InsertDynamicPositionResults(SpecificPipeline(candidatePipeline), dynamicInsertionPosition)

def apply[Query <: PipelineQuery](

candidatePipelines: Set[CandidatePipelineIdentifier],

dynamicInsertionPosition: DynamicInsertionPosition[Query]

): InsertDynamicPositionResults[Query] =

new InsertDynamicPositionResults(

SpecificPipelines(candidatePipelines),

dynamicInsertionPosition)

}

/\*\*

\* Compute a position for inserting the candidates into result. If a `None` is returned, the

\* Selector using this would not insert the candidates into the result.

\*/

trait DynamicInsertionPosition[-Query <: PipelineQuery] {

def apply(

query: Query,

remainingCandidates: Seq[CandidateWithDetails],

result: Seq[CandidateWithDetails]

): Option[Int]

}

/\*\*

\* Insert all candidates in a pipeline scope at a 0-indexed dynamic position computed

\* using the provided [[DynamicInsertionPosition]] instance. If the current results are a shorter

\* length than the computed position, then the candidates will be appended to the results.

\* If the [[DynamicInsertionPosition]] returns a `None`, the candidates are not

\* added to the result.

\*/

case class InsertDynamicPositionResults[-Query <: PipelineQuery](

override val pipelineScope: CandidateScope,

dynamicInsertionPosition: DynamicInsertionPosition[Query])

extends Selector[Query] {

override def apply(

query: Query,

remainingCandidates: Seq[CandidateWithDetails],

result: Seq[CandidateWithDetails]

): SelectorResult = {

dynamicInsertionPosition(query, remainingCandidates, result) match {

case Some(position) =>

InsertSelector.insertIntoResultsAtPosition(

position = position,

pipelineScope = pipelineScope,

remainingCandidates = remainingCandidates,

result = result)

case None =>

// When a valid position is not provided, do not insert the candidates.

// Both the remainingCandidates and result are unchanged.

SelectorResult(remainingCandidates = remainingCandidates, result = result)

}

}

}