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.pipeline.PipelineQuery

object InsertPerCandidateDynamicPositionResults {

def apply[Query <: PipelineQuery](

candidatePipeline: CandidatePipelineIdentifier,

candidatePositionInResults: CandidatePositionInResults[Query]

): InsertPerCandidateDynamicPositionResults[Query] =

InsertPerCandidateDynamicPositionResults[Query](

SpecificPipeline(candidatePipeline),

candidatePositionInResults)

def apply[Query <: PipelineQuery](

candidatePipelines: Set[CandidatePipelineIdentifier],

candidatePositionInResults: CandidatePositionInResults[Query]

): InsertPerCandidateDynamicPositionResults[Query] =

InsertPerCandidateDynamicPositionResults[Query](

SpecificPipelines(candidatePipelines),

candidatePositionInResults)

}

/\*\*

\* Insert each candidate in the [[CandidateScope]] at the index relative to the original candidate in the `result`

\* at that index using the provided [[CandidatePositionInResults]] instance. If the current results are shorter

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

\*

\* When the [[CandidatePositionInResults]] returns a `None`, that candidate is not

\* added to the result. Negative position values are treated as 0 (front of the results).

\*

\* @example if [[CandidatePositionInResults]] results in a candidate mapping from index to candidate of

\* `{0 -> a, 0 -> b, 0 -> c, 1 -> e, 2 -> g, 2 -> h} ` with original `results` = `[D, F]`,

\* then the resulting output would look like `[a, b, c, D, e, F, g, h]`

\*/

case class InsertPerCandidateDynamicPositionResults[-Query <: PipelineQuery](

pipelineScope: CandidateScope,

candidatePositionInResults: CandidatePositionInResults[Query])

extends Selector[Query] {

override def apply(

query: Query,

remainingCandidates: Seq[CandidateWithDetails],

result: Seq[CandidateWithDetails]

): SelectorResult = {

val (candidatesToInsert, otherRemainingCandidatesTuples) = remainingCandidates

.map { candidate: CandidateWithDetails =>

val position =

if (pipelineScope.contains(candidate))

candidatePositionInResults(query, candidate, result)

else

None

(position, candidate)

}.partition { case (index, \_) => index.isDefined }

val otherRemainingCandidates = otherRemainingCandidatesTuples.map {

case (\_, candidate) => candidate

}

val positionAndCandidateList = candidatesToInsert.collect {

case (Some(position), candidate) => (position, candidate)

}

val mergedResult = DynamicPositionSelector.mergeByIndexIntoResult(

positionAndCandidateList,

result,

DynamicPositionSelector.RelativeIndices

)

SelectorResult(remainingCandidates = otherRemainingCandidates, result = mergedResult)

}

}