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

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

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

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

import com.twitter.timelines.configapi.BoundedParam

/\*\*

\* Limit the number of results to min(PipelineQuery.requestedMaxResults, ServerMaxResultsParam)

\*

\* PipelineQuery.requestedMaxResults is optionally set in the pipelineQuery.

\* If it is not set, then the default value of DefaultRequestedMaxResultsParam is used.

\*

\* ServerMaxResultsParam specifies the maximum number of results supported, irrespective of what is

\* specified by the client in PipelineQuery.requestedMaxResults

\* (or the DefaultRequestedMaxResultsParam default if not specified)

\*

\* For example, if ServerMaxResultsParam is 5, PipelineQuery.requestedMaxResults is 3,

\* and the results contain 10 items, then these items will be reduced to the first 3 selected items.

\*

\* If PipelineQuery.requestedMaxResults is not set, DefaultRequestedMaxResultsParam is 3,

\* ServerMaxResultsParam is 5 and the results contain 10 items,

\* then these items will be reduced to the first 3 selected items.

\*

\* Another example, if ServerMaxResultsParam is 5, PipelineQuery.requestedMaxResults is 8,

\* and the results contain 10 items, then these will be reduced to the first 5 selected items.

\*

\* The items inside the modules will not be affected by this selector.

\*/

case class DropRequestedMaxResults(

defaultRequestedMaxResultsParam: BoundedParam[Int],

serverMaxResultsParam: BoundedParam[Int])

extends Selector[PipelineQuery] {

override val pipelineScope: CandidateScope = AllPipelines

override def apply(

query: PipelineQuery,

remainingCandidates: Seq[CandidateWithDetails],

result: Seq[CandidateWithDetails]

): SelectorResult = {

val requestedMaxResults = query.maxResults(defaultRequestedMaxResultsParam)

val serverMaxResults = query.params(serverMaxResultsParam)

assert(requestedMaxResults > 0, "Requested max results must be greater than zero")

assert(serverMaxResults > 0, "Server max results must be greater than zero")

val appliedMaxResults = Math.min(requestedMaxResults, serverMaxResults)

val resultUpdated = DropSelector.takeUntil(appliedMaxResults, result)

SelectorResult(remainingCandidates = remainingCandidates, result = resultUpdated)

}

}