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.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

import com.twitter.timelines.configapi.Param

/\*\*

\* Limit the number of results (for 1 or more modules) from a certain candidate

\* source to PipelineQuery.requestedMaxResults.

\*

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

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

\*

\* For example, if PipelineQuery.requestedMaxResults is 3, and a candidatePipeline returned 1 module

\* containing 10 items in the candidate pool, then these module items will be reduced to the first 3

\* module items. Note that to update the ordering of the candidates, an

\* UpdateModuleItemsCandidateOrderingSelector may be used prior to using this selector.

\*

\* Another example, if PipelineQuery.requestedMaxResults is 3, and a candidatePipeline returned 5

\* modules each containing 10 items in the candidate pool, then the module items in each of the 5

\* modules will be reduced to the first 3 module items.

\*

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

\*/

case class DropRequestedMaxModuleItemCandidates(

override val pipelineScope: CandidateScope,

defaultRequestedMaxModuleItemResultsParam: Param[Int])

extends Selector[PipelineQuery] {

override def apply(

query: PipelineQuery,

remainingCandidates: Seq[CandidateWithDetails],

result: Seq[CandidateWithDetails]

): SelectorResult = {

val requestedMaxModuleItemSelections =

query.maxResults(defaultRequestedMaxModuleItemResultsParam)

assert(

requestedMaxModuleItemSelections > 0,

"Requested Max module item selections must be greater than zero")

val resultUpdated = result.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 =

DropSelector.takeUntil(requestedMaxModuleItemSelections, module.candidates))

case candidate => candidate

}

SelectorResult(remainingCandidates = remainingCandidates, result = resultUpdated)

}

}

object DropRequestedMaxModuleItemCandidates {

def apply(

candidatePipeline: CandidatePipelineIdentifier,

defaultRequestedMaxModuleItemResultsParam: Param[Int]

) =

new DropRequestedMaxModuleItemCandidates(

SpecificPipeline(candidatePipeline),

defaultRequestedMaxModuleItemResultsParam)

}