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

import com.twitter.timelines.configapi.Param

trait MaxSelector[-Query <: PipelineQuery] {

def apply(

query: Query,

remainingCandidates: Seq[CandidateWithDetails],

result: Seq[CandidateWithDetails]

): Int

}

object DropMaxCandidates {

/\*\*

\* A [[DropMaxCandidates]] Selector based on a [[Param]] applied to a single candidate pipeline

\*/

def apply[Query <: PipelineQuery](

candidatePipeline: CandidatePipelineIdentifier,

maxSelectionsParam: Param[Int]

) = new DropMaxCandidates[Query](

SpecificPipeline(candidatePipeline),

(query, \_, \_) => query.params(maxSelectionsParam))

/\*\*

\* A [[DropMaxCandidates]] Selector based on a [[Param]] with multiple candidate pipelines

\*/

def apply[Query <: PipelineQuery](

candidatePipelines: Set[CandidatePipelineIdentifier],

maxSelectionsParam: Param[Int]

) = new DropMaxCandidates[Query](

SpecificPipelines(candidatePipelines),

(query, \_, \_) => query.params(maxSelectionsParam))

/\*\*

\* A [[DropMaxCandidates]] Selector based on a [[Param]] that applies to a [[CandidateScope]]

\*/

def apply[Query <: PipelineQuery](

pipelineScope: CandidateScope,

maxSelectionsParam: Param[Int]

) = new DropMaxCandidates[Query](pipelineScope, (query, \_, \_) => query.params(maxSelectionsParam))

}

/\*\*

\* Limit the number of item and module (not items inside modules) candidates from the

\* specified pipelines based on the value provided by the [[MaxSelector]]

\*

\* For example, if value from the [[MaxSelector]] is 3, and a candidatePipeline returned 10 items

\* in the candidate pool, then these items will be reduced to the first 3 items. Note that to

\* update the ordering of the candidates, an UpdateCandidateOrderingSelector may be used prior to

\* using this Selector.

\*

\* Another example, if the [[MaxSelector]] value is 3, and a candidatePipeline returned 10 modules

\* in the candidate pool, then these will be reduced to the first 3 modules. The items inside the

\* modeles will not be affected by this selector. To control the number of items inside modules see

\* [[DropMaxModuleItemCandidates]].

\*/

case class DropMaxCandidates[-Query <: PipelineQuery](

override val pipelineScope: CandidateScope,

maxSelector: MaxSelector[Query])

extends Selector[Query] {

override def apply(

query: Query,

remainingCandidates: Seq[CandidateWithDetails],

result: Seq[CandidateWithDetails]

): SelectorResult = {

val maxSelections = maxSelector(query, remainingCandidates, result)

assert(maxSelections > 0, "Max selections must be greater than zero")

val remainingCandidatesLimited =

DropSelector.takeUntil(maxSelections, remainingCandidates, pipelineScope)

SelectorResult(remainingCandidates = remainingCandidatesLimited, result = result)

}

}