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

import com.twitter.product\_mixer.component\_library.selector.DropSelector.dropDuplicates

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

/\*\*

\* Keep only the first instance of a candidate in the `result` as determined by comparing

\* the contained candidate ID and class type. Subsequent matching instances will be dropped. For

\* more details, see DropSelector#dropDuplicates

\*

\* @param duplicationKey how to generate the key used to identify duplicate candidates (by default use id and class name)

\* @param mergeStrategy how to merge two candidates with the same key (by default pick the first one)

\*

\* @note [[com.twitter.product\_mixer.component\_library.model.candidate.CursorCandidate]] are ignored.

\* @note [[com.twitter.product\_mixer.core.model.common.presentation.ModuleCandidateWithDetails]] are ignored.

\*

\* @example if `result`

\* `Seq(sourceA\_Id1, sourceA\_Id1, sourceA\_Id2, sourceB\_id1, sourceB\_id2, sourceB\_id3, sourceC\_id4)`

\* then the output result will be `Seq(sourceA\_Id1, sourceA\_Id2, sourceB\_id3, sourceC\_id4)`

\*/

case class DropDuplicateResults(

duplicationKey: DeduplicationKey[\_] = IdAndClassDuplicationKey,

mergeStrategy: CandidateMergeStrategy = PickFirstCandidateMerger)

extends Selector[PipelineQuery] {

override val pipelineScope: CandidateScope = AllPipelines

override def apply(

query: PipelineQuery,

remainingCandidates: Seq[CandidateWithDetails],

result: Seq[CandidateWithDetails]

): SelectorResult = {

val dedupedResults = dropDuplicates(

pipelineScope = pipelineScope,

candidates = result,

duplicationKey = duplicationKey,

mergeStrategy = mergeStrategy)

SelectorResult(remainingCandidates = remainingCandidates, result = dedupedResults)

}

}