package com.twitter.product\_mixer.core.functional\_component.common

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.CandidatePipelines

/\*\*

\* Specifies whether a function component (e.g, [[Gate]] or [[Selector]])

\* should apply to a given [[CandidateWithDetails]]

\*/

sealed trait CandidateScope {

/\*\*

\* returns True if the provided `candidate` is in scope

\*/

def contains(candidate: CandidateWithDetails): Boolean

/\*\* partitions `candidates` into those that this scope [[contains]] and those it does not \*/

final def partition(

candidates: Seq[CandidateWithDetails]

): CandidateScope.PartitionedCandidates = {

val (candidatesInScope, candidatesOutOfScope) = candidates.partition(contains)

CandidateScope.PartitionedCandidates(candidatesInScope, candidatesOutOfScope)

}

}

object CandidateScope {

case class PartitionedCandidates(

candidatesInScope: Seq[CandidateWithDetails],

candidatesOutOfScope: Seq[CandidateWithDetails])

}

/\*\*

\* A [[CandidateScope]] that applies the given functional component

\* to all candidates regardless of which pipeline is their [[com.twitter.product\_mixer.core.model.common.presentation.CandidateWithDetails.source]].

\*/

case object AllPipelines extends CandidateScope {

override def contains(candidate: CandidateWithDetails): Boolean = true

}

/\*\*

\* A [[CandidateScope]] that applies the given [[com.twitter.product\_mixer.core.functional\_component.selector.Selector]]

\* only to candidates whose [[com.twitter.product\_mixer.core.model.common.presentation.CandidatePipelines]]

\* has an Identifier in the [[pipelines]] Set.

\* In most cases where candidates are not pre-merged, the Set contains the candidate pipeline identifier the candidate

\* came from. In the case where a candidate's feature maps were merged using [[CombineFeatureMapsCandidateMerger]], the

\* set contains all candidate pipelines the merged candidate came from and this scope will include the candidate if any

\* of the pipelines match.

\*/

case class SpecificPipelines(pipelines: Set[CandidatePipelineIdentifier]) extends CandidateScope {

require(

pipelines.nonEmpty,

"Expected `SpecificPipelines` have a non-empty Set of CandidatePipelineIdentifiers.")

override def contains(candidate: CandidateWithDetails): Boolean = {

candidate.features.get(CandidatePipelines).exists(pipelines.contains)

}

}

/\*\*

\* A [[CandidateScope]] that applies the given [[com.twitter.product\_mixer.core.functional\_component.selector.Selector]]

\* only to candidates whose [[com.twitter.product\_mixer.core.model.common.presentation.CandidateWithDetails.source]]

\* is [[pipeline]].

\*/

case class SpecificPipeline(pipeline: CandidatePipelineIdentifier) extends CandidateScope {

override def contains(candidate: CandidateWithDetails): Boolean = candidate.features

.get(CandidatePipelines).contains(pipeline)

}

object SpecificPipelines {

def apply(

pipeline: CandidatePipelineIdentifier,

pipelines: CandidatePipelineIdentifier\*

): CandidateScope = {

if (pipelines.isEmpty)

SpecificPipeline(pipeline)

else

SpecificPipelines((pipeline +: pipelines).toSet)

}

}

/\*\*

\* A [[CandidateScope]] that applies the given [[com.twitter.product\_mixer.core.functional\_component.selector.Selector]]

\* to all candidates except for the candidates whose [[com.twitter.product\_mixer.core.model.common.presentation.CandidatePipelines]]

\* has an Identifier in the [[pipelinesToExclude]] Set.

\* In most cases where candidates are not pre-merged, the Set contains the candidate pipeline identifier the candidate

\* came from. In the case where a candidate's feature maps were merged using [[CombineFeatureMapsCandidateMerger]], the

\* set contains all candidate pipelines the merged candidate came from and this scope will include the candidate if any

\* of the pipelines match.

\*/

case class AllExceptPipelines(

pipelinesToExclude: Set[CandidatePipelineIdentifier])

extends CandidateScope {

override def contains(candidate: CandidateWithDetails): Boolean = !candidate.features

.get(CandidatePipelines).exists(pipelinesToExclude.contains)

}