package com.twitter.product\_mixer.core.pipeline.candidate

import com.twitter.product\_mixer.core.feature.featuremap.FeatureMap

import com.twitter.product\_mixer.core.feature.featuremap.asyncfeaturemap.AsyncFeatureMap

import com.twitter.product\_mixer.core.model.common.UniversalNoun

import com.twitter.product\_mixer.core.model.common.identifier.CandidateSourceIdentifier

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

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

import com.twitter.product\_mixer.core.pipeline.pipeline\_failure.PipelineFailure

import com.twitter.product\_mixer.core.service.async\_feature\_map\_executor.AsyncFeatureMapExecutorResults

import com.twitter.product\_mixer.core.service.candidate\_decorator\_executor.CandidateDecoratorExecutorResult

import com.twitter.product\_mixer.core.service.candidate\_feature\_hydrator\_executor.CandidateFeatureHydratorExecutorResult

import com.twitter.product\_mixer.core.service.candidate\_source\_executor.CandidateSourceExecutorResult

import com.twitter.product\_mixer.core.service.filter\_executor.FilterExecutorResult

import com.twitter.product\_mixer.core.service.gate\_executor.GateExecutorResult

import com.twitter.product\_mixer.core.service.query\_feature\_hydrator\_executor.QueryFeatureHydratorExecutor

case class CandidatePipelineResult(

candidateSourceIdentifier: CandidateSourceIdentifier,

gateResult: Option[GateExecutorResult],

queryFeatures: Option[QueryFeatureHydratorExecutor.Result],

queryFeaturesPhase2: Option[QueryFeatureHydratorExecutor.Result],

mergedAsyncQueryFeatures: Option[AsyncFeatureMap],

candidateSourceResult: Option[CandidateSourceExecutorResult[UniversalNoun[Any]]],

preFilterHydrationResult: Option[CandidateFeatureHydratorExecutorResult[UniversalNoun[Any]]],

preFilterHydrationResultPhase2: Option[

CandidateFeatureHydratorExecutorResult[UniversalNoun[Any]]

],

filterResult: Option[FilterExecutorResult[UniversalNoun[Any]]],

postFilterHydrationResult: Option[CandidateFeatureHydratorExecutorResult[UniversalNoun[Any]]],

candidateDecoratorResult: Option[CandidateDecoratorExecutorResult],

scorersResult: Option[CandidateFeatureHydratorExecutorResult[UniversalNoun[Any]]],

asyncFeatureHydrationResults: Option[AsyncFeatureMapExecutorResults],

failure: Option[PipelineFailure],

result: Option[Seq[CandidateWithDetails]])

extends PipelineResult[Seq[CandidateWithDetails]] {

override def withFailure(failure: PipelineFailure): CandidatePipelineResult =

copy(failure = Some(failure))

override def withResult(

result: Seq[CandidateWithDetails]

): CandidatePipelineResult = copy(result = Some(result))

override val resultSize: Int = result.map(PipelineResult.resultSize).getOrElse(0)

}

private[candidate] object IntermediateCandidatePipelineResult {

def empty[Candidate <: UniversalNoun[Any]](

candidateSourceIdentifier: CandidateSourceIdentifier

): IntermediateCandidatePipelineResult[Candidate] = {

IntermediateCandidatePipelineResult(

CandidatePipelineResult(

candidateSourceIdentifier = candidateSourceIdentifier,

None,

None,

None,

None,

None,

None,

None,

None,

None,

None,

None,

None,

None,

None

),

None

)

}

}

private[candidate] case class IntermediateCandidatePipelineResult[Candidate <: UniversalNoun[Any]](

underlyingResult: CandidatePipelineResult,

featureMaps: Option[Map[Candidate, FeatureMap]])

extends PipelineResult[Seq[CandidateWithDetails]] {

override val failure: Option[PipelineFailure] = underlyingResult.failure

override val result: Option[Seq[CandidateWithDetails]] = underlyingResult.result

override def withFailure(

failure: PipelineFailure

): IntermediateCandidatePipelineResult[Candidate] =

copy(underlyingResult = underlyingResult.withFailure(failure))

override def withResult(

result: Seq[CandidateWithDetails]

): IntermediateCandidatePipelineResult[Candidate] =

copy(underlyingResult = underlyingResult.withResult(result))

override def resultSize(): Int = underlyingResult.resultSize

}