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

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.marshalling.HasMarshalling

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\_pipeline\_executor.CandidatePipelineExecutorResult

import com.twitter.product\_mixer.core.service.domain\_marshaller\_executor.DomainMarshallerExecutor

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.pipeline\_result\_side\_effect\_executor.PipelineResultSideEffectExecutor

import com.twitter.product\_mixer.core.service.quality\_factor\_executor.QualityFactorExecutorResult

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

import com.twitter.product\_mixer.core.service.scoring\_pipeline\_executor.ScoringPipelineExecutorResult

import com.twitter.product\_mixer.core.service.selector\_executor.SelectorExecutorResult

import com.twitter.product\_mixer.core.service.transport\_marshaller\_executor.TransportMarshallerExecutor

case class RecommendationPipelineResult[Candidate <: UniversalNoun[Any], ResultType](

qualityFactorResult: Option[QualityFactorExecutorResult],

gateResult: Option[GateExecutorResult],

queryFeatures: Option[QueryFeatureHydratorExecutor.Result],

queryFeaturesPhase2: Option[QueryFeatureHydratorExecutor.Result],

mergedAsyncQueryFeatures: Option[AsyncFeatureMap],

candidatePipelineResults: Option[CandidatePipelineExecutorResult],

dependentCandidatePipelineResults: Option[CandidatePipelineExecutorResult],

postCandidatePipelinesSelectorResults: Option[SelectorExecutorResult],

postCandidatePipelinesFeatureHydrationResults: Option[

CandidateFeatureHydratorExecutorResult[Candidate]

],

globalFilterResults: Option[FilterExecutorResult[Candidate]],

scoringPipelineResults: Option[ScoringPipelineExecutorResult[Candidate]],

resultSelectorResults: Option[SelectorExecutorResult],

postSelectionFilterResults: Option[FilterExecutorResult[Candidate]],

candidateDecoratorResult: Option[CandidateDecoratorExecutorResult],

domainMarshallerResults: Option[DomainMarshallerExecutor.Result[HasMarshalling]],

resultSideEffectResults: Option[PipelineResultSideEffectExecutor.Result],

asyncFeatureHydrationResults: Option[AsyncFeatureMapExecutorResults],

transportMarshallerResults: Option[TransportMarshallerExecutor.Result[ResultType]],

failure: Option[PipelineFailure],

result: Option[ResultType])

extends PipelineResult[ResultType] {

override val resultSize: Int = result match {

case Some(seqResult @ Seq(\_)) => seqResult.length

case Some(\_) => 1

case None => 0

}

override def withFailure(

failure: PipelineFailure

): RecommendationPipelineResult[Candidate, ResultType] =

copy(failure = Some(failure))

override def withResult(result: ResultType): RecommendationPipelineResult[Candidate, ResultType] =

copy(result = Some(result))

}

object RecommendationPipelineResult {

def empty[A <: UniversalNoun[Any], B]: RecommendationPipelineResult[A, B] =

RecommendationPipelineResult(

None,

None,

None,

None,

None,

None,

None,

None,

None,

None,

None,

None,

None,

None,

None,

None,

None,

None,

None,

None

)

}