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

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

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\_pipeline\_executor.CandidatePipelineExecutorResult

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

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.selector\_executor.SelectorExecutorResult

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

/\*\*

\* A [[MixerPipelineResult]] includes both the user-visible [[PipelineResult]] and all the

\* Execution details possible - intermediate results, what components did, etc.

\*/

case class MixerPipelineResult[Result](

qualityFactorResult: Option[QualityFactorExecutorResult],

gateResult: Option[GateExecutorResult],

queryFeatures: Option[QueryFeatureHydratorExecutor.Result],

queryFeaturesPhase2: Option[QueryFeatureHydratorExecutor.Result],

mergedAsyncQueryFeatures: Option[AsyncFeatureMap],

candidatePipelineResults: Option[CandidatePipelineExecutorResult],

dependentCandidatePipelineResults: Option[CandidatePipelineExecutorResult],

resultSelectorResults: Option[SelectorExecutorResult],

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

resultSideEffectResults: Option[PipelineResultSideEffectExecutor.Result],

asyncFeatureHydrationResults: Option[AsyncFeatureMapExecutorResults],

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

failure: Option[PipelineFailure],

result: Option[Result])

extends PipelineResult[Result] {

override def withFailure(failure: PipelineFailure): PipelineResult[Result] =

copy(failure = Some(failure))

override def withResult(result: Result): PipelineResult[Result] = copy(result = Some(result))

/\*\*

\* resultSize is calculated based on the selector results rather than the marshalled results. The

\* structure of the marshalled format is unknown, making operating on selector results more

\* convenient. This will implicitly excluded cursors built during marshalling but cursors don't

\* contribute to the result size anyway.

\*/

override val resultSize: Int =

resultSelectorResults.map(\_.selectedCandidates).map(PipelineResult.resultSize).getOrElse(0)

}

object MixerPipelineResult {

def empty[A]: MixerPipelineResult[A] = MixerPipelineResult(

None,

None,

None,

None,

None,

None,

None,

None,

None,

None,

None,

None,

None,

None

)

}