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

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

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

import com.twitter.product\_mixer.core.pipeline.mixer.MixerPipelineResult

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

import com.twitter.product\_mixer.core.pipeline.recommendation.RecommendationPipelineResult

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

import com.twitter.product\_mixer.core.service.pipeline\_selector\_executor.PipelineSelectorExecutorResult

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

case class ProductPipelineResult[Result](

transformedQuery: Option[PipelineQuery],

qualityFactorResult: Option[QualityFactorExecutorResult],

gateResult: Option[GateExecutorResult],

pipelineSelectorResult: Option[PipelineSelectorExecutorResult],

mixerPipelineResult: Option[MixerPipelineResult[Result]],

recommendationPipelineResult: Option[RecommendationPipelineResult[\_, Result]],

traceId: Option[String],

failure: Option[PipelineFailure],

result: Option[Result])

extends PipelineResult[Result] {

override val resultSize: Int = {

if (mixerPipelineResult.isDefined) {

mixerPipelineResult.map(\_.resultSize).getOrElse(0)

} else {

recommendationPipelineResult.map(\_.resultSize).getOrElse(0)

}

}

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

copy(failure = Some(failure))

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

}

object ProductPipelineResult {

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

None,

None,

None,

None,

None,

None,

None,

None,

None

)

def fromResult[A](result: A): ProductPipelineResult[A] = ProductPipelineResult(

None,

None,

None,

None,

None,

None,

None,

None,

Some(result)

)

}