package com.twitter.product\_mixer.core.functional\_component.side\_effect

import com.twitter.product\_mixer.core.functional\_component.side\_effect.PipelineResultSideEffect.Inputs

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

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

import com.twitter.product\_mixer.core.model.marshalling.HasMarshalling

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

/\*\*

\* A side-effect that can be run with a pipeline result before transport marshalling

\*

\* @see SideEffect

\*

\* @tparam Query pipeline query

\* @tparam ResultType response after domain marshalling

\*/

trait PipelineResultSideEffect[-Query <: PipelineQuery, -ResultType <: HasMarshalling]

extends SideEffect[Inputs[Query, ResultType]]

with PipelineResultSideEffect.SupportsConditionally[Query, ResultType]

object PipelineResultSideEffect {

/\*\*

\* Mixin for when you want to conditionally run a [[PipelineResultSideEffect]]

\*

\* This is a thin wrapper around [[common.Conditionally]] exposing a nicer API for the [[PipelineResultSideEffect]] specific use-case.

\*/

trait Conditionally[-Query <: PipelineQuery, -ResultType <: HasMarshalling]

extends common.Conditionally[Inputs[Query, ResultType]] {

\_: PipelineResultSideEffect[Query, ResultType] =>

/\*\* @see [[common.Conditionally.onlyIf]] \*/

def onlyIf(

query: Query,

selectedCandidates: Seq[CandidateWithDetails],

remainingCandidates: Seq[CandidateWithDetails],

droppedCandidates: Seq[CandidateWithDetails],

response: ResultType

): Boolean

override final def onlyIf(input: Inputs[Query, ResultType]): Boolean =

onlyIf(

input.query,

input.selectedCandidates,

input.remainingCandidates,

input.droppedCandidates,

input.response)

}

type SupportsConditionally[-Query <: PipelineQuery, -ResultType <: HasMarshalling] =

common.SupportsConditionally[Inputs[Query, ResultType]]

case class Inputs[+Query <: PipelineQuery, +ResultType <: HasMarshalling](

query: Query,

selectedCandidates: Seq[CandidateWithDetails],

remainingCandidates: Seq[CandidateWithDetails],

droppedCandidates: Seq[CandidateWithDetails],

response: ResultType)

}