package com.twitter.product\_mixer.core.pipeline.step.domain\_marshaller

import com.twitter.product\_mixer.core.functional\_component.premarshaller.DomainMarshaller

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

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

import com.twitter.product\_mixer.core.pipeline.state.HasCandidatesWithDetails

import com.twitter.product\_mixer.core.pipeline.state.HasQuery

import com.twitter.product\_mixer.core.pipeline.step.Step

import com.twitter.product\_mixer.core.service.Executor

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

import com.twitter.stitch.Arrow

import javax.inject.Inject

/\*\*

\* A domain marshaller step, it takes the input list of candidates and the given

\* domain marshaller and executes its to return a marshalled result. The [[State]] object is

\* responsible for keeping a reference of the built Response.

\*

\* @param domainMarshallerExecutor Domain Marshaller executor.

\* @tparam Query Type of PipelineQuery domain model

\* @tparam ResponseType the domain marshalling type expected to be returned.

\* @tparam State The pipeline state domain model.

\*/

case class DomainMarshallerStep[

Query <: PipelineQuery,

ResponseType <: HasMarshalling,

State <: HasQuery[Query, State] with HasCandidatesWithDetails[State]] @Inject() (

domainMarshallerExecutor: DomainMarshallerExecutor)

extends Step[State, DomainMarshaller[Query, ResponseType], DomainMarshallerExecutor.Inputs[

Query

], DomainMarshallerExecutor.Result[ResponseType]] {

override def isEmpty(config: DomainMarshaller[Query, ResponseType]): Boolean = false

override def adaptInput(

state: State,

config: DomainMarshaller[Query, ResponseType]

): DomainMarshallerExecutor.Inputs[Query] =

DomainMarshallerExecutor.Inputs(state.query, state.candidatesWithDetails)

override def arrow(

config: DomainMarshaller[Query, ResponseType],

context: Executor.Context

): Arrow[DomainMarshallerExecutor.Inputs[Query], DomainMarshallerExecutor.Result[ResponseType]] =

domainMarshallerExecutor.arrow(config, context)

// Noop since the pipeline updates the executor results for us

override def updateState(

state: State,

executorResult: DomainMarshallerExecutor.Result[ResponseType],

config: DomainMarshaller[Query, ResponseType]

): State = state

}