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

import com.twitter.product\_mixer.core.functional\_component.decorator.CandidateDecorator

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

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

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.HasCandidatesWithFeatures

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.candidate\_decorator\_executor.CandidateDecoratorExecutor

import com.twitter.product\_mixer.core.service.candidate\_decorator\_executor.CandidateDecoratorExecutorResult

import com.twitter.stitch.Arrow

import javax.inject.Inject

/\*\*

\* A candidate decoration step, which takes the query and candidates and outputs decorations for them

\*

\* @param candidateDecoratorExecutor Candidate Source Executor

\* @tparam Query Type of PipelineQuery domain model

\* @tparam Candidate Type of Candidates to filter

\* @tparam State The pipeline state domain model.

\*/

case class DecoratorStep[

Query <: PipelineQuery,

Candidate <: UniversalNoun[Any],

State <: HasQuery[Query, State] with HasCandidatesWithDetails[

State

] with HasCandidatesWithFeatures[

Candidate,

State

]] @Inject() (candidateDecoratorExecutor: CandidateDecoratorExecutor)

extends Step[

State,

Option[CandidateDecorator[Query, Candidate]],

(Query, Seq[CandidateWithFeatures[Candidate]]),

CandidateDecoratorExecutorResult

] {

override def isEmpty(config: Option[CandidateDecorator[Query, Candidate]]): Boolean =

config.isEmpty

override def adaptInput(

state: State,

config: Option[CandidateDecorator[Query, Candidate]]

): (Query, Seq[CandidateWithFeatures[Candidate]]) =

(state.query, state.candidatesWithFeatures)

override def arrow(

config: Option[CandidateDecorator[Query, Candidate]],

context: Executor.Context

): Arrow[(Query, Seq[CandidateWithFeatures[Candidate]]), CandidateDecoratorExecutorResult] =

candidateDecoratorExecutor.arrow(config, context)

override def updateState(

state: State,

executorResult: CandidateDecoratorExecutorResult,

config: Option[CandidateDecorator[Query, Candidate]]

): State = {

state.updateDecorations(executorResult.result)

}

}