package com.twitter.product\_mixer.core.service.candidate\_decorator\_executor

import com.twitter.finagle.stats.StatsReceiver

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

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

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.service.Executor

import com.twitter.stitch.Arrow

import javax.inject.Inject

import javax.inject.Singleton

@Singleton

class CandidateDecoratorExecutor @Inject() (override val statsReceiver: StatsReceiver)

extends Executor {

def arrow[Query <: PipelineQuery, Candidate <: UniversalNoun[Any]](

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

context: Executor.Context

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

val decoratorArrow =

decoratorOpt match {

case Some(decorator) =>

val candidateDecoratorArrow =

Arrow.flatMap[(Query, Seq[CandidateWithFeatures[Candidate]]), Seq[Decoration]] {

case (query, candidatesWithFeatures) => decorator.apply(query, candidatesWithFeatures)

}

wrapComponentWithExecutorBookkeeping(context, decorator.identifier)(

candidateDecoratorArrow)

case \_ => Arrow.value(Seq.empty[Decoration])

}

decoratorArrow.map(CandidateDecoratorExecutorResult)

}

}