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

import com.twitter.product\_mixer.core.functional\_component.candidate\_source.PassthroughCandidateSource

import com.twitter.product\_mixer.core.functional\_component.candidate\_source.CandidateExtractor

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

import com.twitter.product\_mixer.core.functional\_component.transformer.CandidatePipelineQueryTransformer

import com.twitter.product\_mixer.core.functional\_component.transformer.CandidatePipelineResultsTransformer

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

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

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

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

object PassthroughCandidatePipelineConfig {

/\*\*

\* Build a [[PassthroughCandidatePipelineConfig]] with a [[PassthroughCandidateSource]] built from

\* a [[CandidateExtractor]]

\*/

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

identifier: CandidatePipelineIdentifier,

extractor: CandidateExtractor[Query, Candidate],

decorator: Option[CandidateDecorator[Query, Candidate]] = None

): PassthroughCandidatePipelineConfig[Query, Candidate] = {

// Renaming variables to keep the interface clean, but avoid naming collisions when creating

// the anonymous class.

val \_identifier = identifier

val \_extractor = extractor

val \_decorator = decorator

new PassthroughCandidatePipelineConfig[Query, Candidate] {

override val identifier = \_identifier

override val candidateSource =

PassthroughCandidateSource(CandidateSourceIdentifier(\_identifier.name), \_extractor)

override val decorator = \_decorator

}

}

}

trait PassthroughCandidatePipelineConfig[Query <: PipelineQuery, Candidate <: UniversalNoun[Any]]

extends CandidatePipelineConfig[Query, Query, Candidate, Candidate] {

override val queryTransformer: CandidatePipelineQueryTransformer[Query, Query] = identity

override val resultTransformer: CandidatePipelineResultsTransformer[Candidate, Candidate] =

identity

}