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

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

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

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

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

object StaticCandidatePipelineConfig {

/\*\*

\* Build a [[StaticCandidatePipelineConfig]] with a [[CandidateSource]] that returns the [[candidate]]

\*/

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

identifier: CandidatePipelineIdentifier,

candidate: Candidate,

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

): StaticCandidatePipelineConfig[Query, Candidate] = {

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

// the anonymous class.

val \_identifier = identifier

val \_candidate = candidate

val \_decorator = decorator

new StaticCandidatePipelineConfig[Query, Candidate] {

override val identifier = \_identifier

override val candidate = \_candidate

override val decorator = \_decorator

}

}

}

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

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

val candidate: Candidate

override def candidateSource: CandidateSource[Unit, Unit] = StaticCandidateSource[Unit](

identifier = CandidateSourceIdentifier(identifier.name),

result = Seq(()))

override val queryTransformer: CandidatePipelineQueryTransformer[Query, Unit] = \_ => Unit

override val resultTransformer: CandidatePipelineResultsTransformer[Unit, Candidate] = \_ =>

candidate

}