package com.twitter.product\_mixer.component\_library.pipeline.candidate.flexible\_injection\_pipeline

import com.twitter.product\_mixer.component\_library.candidate\_source.flexible\_injection\_pipeline.PromptCandidateSource

import com.twitter.product\_mixer.component\_library.pipeline.candidate.flexible\_injection\_pipeline.transformer.HasFlipInjectionParams

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

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

import com.twitter.timelines.configapi.FSParam

import com.twitter.timelines.configapi.decider.DeciderParam

import javax.inject.Inject

import javax.inject.Singleton

@Singleton

class FlipPromptDependentCandidatePipelineConfigBuilder @Inject() (

promptCandidateSource: PromptCandidateSource) {

/\*\*

\* Build a FlipPromptDependentCandidatePipelineConfig

\*

\* @note If injected classes are needed to populate parameters in this method, consider creating a

\* ProductFlipPromptDependentCandidatePipelineConfigBuilder with a single `def build()` method.

\* That product-specific builder class can then inject everything it needs (including this

\* class), and delegate to this class's build() method within its own build() method.

\*/

def build[Query <: PipelineQuery with HasFlipInjectionParams](

identifier: CandidatePipelineIdentifier = CandidatePipelineIdentifier("FlipPrompt"),

enabledDeciderParam: Option[DeciderParam[Boolean]] = None,

supportedClientParam: Option[FSParam[Boolean]] = None,

): FlipPromptDependentCandidatePipelineConfig[Query] = {

new FlipPromptDependentCandidatePipelineConfig(

identifier = identifier,

enabledDeciderParam = enabledDeciderParam,

supportedClientParam = supportedClientParam,

promptCandidateSource = promptCandidateSource)

}

}