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

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

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

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.ExecutorResult

import com.twitter.stitch.Arrow

import javax.inject.Inject

/\*\*

\* Pipeline Selection step to decide which pipeline to execute. This step doesn't update state, as

\* the selected pipeline identifier is added to the executor results list map for later retrieval

\*

\* @tparam Query Pipeline query model

\* @tparam State The pipeline state domain model.

\*/

case class PipelineSelectorStep[Query <: PipelineQuery, State <: HasQuery[Query, State]] @Inject() (

) extends Step[State, Query => ComponentIdentifier, Query, PipelineSelectorResult] {

override def isEmpty(config: Query => ComponentIdentifier): Boolean = false

override def adaptInput(

state: State,

config: Query => ComponentIdentifier

): Query = state.query

override def arrow(

config: Query => ComponentIdentifier,

context: Executor.Context

): Arrow[Query, PipelineSelectorResult] = Arrow.map { query: Query =>

PipelineSelectorResult(config(query))

}

// Noop since we keep the identifier in the executor results

override def updateState(

state: State,

executorResult: PipelineSelectorResult,

config: Query => ComponentIdentifier

): State = state

}

case class PipelineSelectorResult(pipelineIdentifier: ComponentIdentifier) extends ExecutorResult