package com.twitter.product\_mixer.core.service.pipeline\_selector\_executor

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.util.logging.Logging

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

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

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

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

import com.twitter.product\_mixer.core.pipeline.pipeline\_failure.InvalidPipelineSelected

import com.twitter.product\_mixer.core.pipeline.pipeline\_failure.PipelineFailure

import com.twitter.product\_mixer.core.service.Executor

import com.twitter.stitch.Arrow

import javax.inject.Inject

import javax.inject.Singleton

@Singleton

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

extends Executor

with Logging {

val identifier: ComponentIdentifier = PlatformIdentifier("PipelineSelector")

def arrow[Query <: PipelineQuery, Response](

pipelineByIdentifier: Map[ComponentIdentifier, Pipeline[Query, Response]],

pipelineSelector: Query => ComponentIdentifier,

context: Executor.Context

): Arrow[Query, PipelineSelectorExecutorResult] = {

val validateSelectedPipelineExists = Arrow

.map(pipelineSelector)

.map { chosenIdentifier =>

if (pipelineByIdentifier.contains(chosenIdentifier)) {

PipelineSelectorExecutorResult(chosenIdentifier)

} else {

// throwing instead of returning a `Throw(\_)` and then `.lowerFromTry` because this is an exceptional case and we want to emphasize that by explicitly throwing

throw PipelineFailure(

InvalidPipelineSelected,

s"${context.componentStack.peek} attempted to select $chosenIdentifier",

// the `componentStack` includes the missing pipeline so it can show up in metrics easier

componentStack = Some(context.componentStack.push(chosenIdentifier))

)

}

}

wrapWithErrorHandling(context, identifier)(validateSelectedPipelineExists)

}

}