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

import com.fasterxml.jackson.databind.annotation.JsonSerialize

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

import scala.util.control.NoStackTrace

/\*\*

\* Pipeline Failures represent pipeline requests that were not able to complete.

\*

\* A pipeline result will always define either a result or a failure.

\*

\* The reason field should not be displayed to end-users, and is free to change over time.

\* It should always be free of private user data such that we can log it.

\*

\* The pipeline can classify it's own failures into categories (timeouts, invalid arguments,

\* rate limited, etc) such that the caller can choose how to handle it.

\*

\* @note [[componentStack]] should only be set by the product mixer framework,

\* it should \*\*NOT\*\* be set when making a [[PipelineFailure]]

\*/

@JsonSerialize(using = classOf[PipelineFailureSerializer])

case class PipelineFailure(

category: PipelineFailureCategory,

reason: String,

underlying: Option[Throwable] = None,

componentStack: Option[ComponentIdentifierStack] = None)

extends Exception(

"PipelineFailure(" +

s"category = $category, " +

s"reason = $reason, " +

s"underlying = $underlying, " +

s"componentStack = $componentStack)",

underlying.orNull

) {

override def toString: String = getMessage

/\*\* Returns an updated copy of this [[PipelineFailure]] with the same exception stacktrace \*/

def copy(

category: PipelineFailureCategory = this.category,

reason: String = this.reason,

underlying: Option[Throwable] = this.underlying,

componentStack: Option[ComponentIdentifierStack] = this.componentStack

): PipelineFailure = {

val newPipelineFailure =

new PipelineFailure(category, reason, underlying, componentStack) with NoStackTrace

newPipelineFailure.setStackTrace(this.getStackTrace)

newPipelineFailure

}

}