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

import com.twitter.product\_mixer.core.model.common.Component

import com.twitter.stitch.Arrow

import com.twitter.stitch.Stitch

/\*\* Base trait for all `pipeline` implementations \*/

trait Pipeline[-Query, Result] extends Component {

/\*\* The [[PipelineConfig]] that was used to create this [[Pipeline]] \*/

private[core] val config: PipelineConfig

/\*\* Returns the underlying arrow that represents the pipeline. This is a val because we want to ensure

\* that the arrow is long-lived and consistent, not generated per-request.

\*

\* Directly using this arrow allows you to combine it with other arrows efficiently.

\*/

val arrow: Arrow[Query, PipelineResult[Result]]

/\*\* all child [[Component]]s that this [[Pipeline]] contains which will be registered and monitored \*/

val children: Seq[Component]

/\*\*

\* A helper for executing a single query.

\*

\* toResultTry and lowerFromTry has the end result of adapting PipelineResult into either a

\* successful result or a Stitch exception, which is a common use-case for callers,

\* particularly in the case of [[com.twitter.product\_mixer.core.pipeline.product.ProductPipeline]].

\*/

def process(query: Query): Stitch[Result] = arrow(query).map(\_.toResultTry).lowerFromTry

final override def toString = s"Pipeline(identifier=$identifier)"

/\*\*

\* [[Pipeline]]s are equal to one another if they were generated from the same [[PipelineConfig]],

\* we check this by doing a reference checks first then comparing the [[PipelineConfig]] instances.

\*

\* We can skip additional checks because the other fields (e.g. [[identifier]] and [[children]])

\* are derived from the [[PipelineConfig]].

\*/

final override def equals(obj: Any): Boolean = obj match {

case pipeline: Pipeline[\_, \_] =>

pipeline.eq(this) || pipeline.config.eq(config) || pipeline.config == config

case \_ => false

}

}