package com.twitter.product\_mixer.core.functional\_component.transformer

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

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

import com.twitter.product\_mixer.core.model.common.presentation.CandidateWithDetails

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

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

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

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

/\*\*

\* A transformer for transforming a mixer or recommendation pipeline's query type into a candidate

\* pipeline's query type.

\* @tparam Query The parent pipeline's query type

\* @tparam CandidateSourceQuery The Candidate Source's query type that the Query should be converted to

\*/

protected[core] sealed trait BaseCandidatePipelineQueryTransformer[

-Query <: PipelineQuery,

+CandidateSourceQuery]

extends Transformer[Query, CandidateSourceQuery] {

override val identifier: TransformerIdentifier =

BaseCandidatePipelineQueryTransformer.DefaultTransformerId

}

trait CandidatePipelineQueryTransformer[-Query <: PipelineQuery, CandidateSourceQuery]

extends BaseCandidatePipelineQueryTransformer[Query, CandidateSourceQuery]

trait DependentCandidatePipelineQueryTransformer[-Query <: PipelineQuery, CandidateSourceQuery]

extends BaseCandidatePipelineQueryTransformer[Query, CandidateSourceQuery] {

def transform(query: Query, candidates: Seq[CandidateWithDetails]): CandidateSourceQuery

final override def transform(query: Query): CandidateSourceQuery = {

val candidates = query.features

.map(\_.get(CandidatePipelineResults)).getOrElse(

throw PipelineFailure(

IllegalStateFailure,

"Candidate Pipeline Results Feature missing from query features"))

transform(query, candidates)

}

}

object BaseCandidatePipelineQueryTransformer {

private[core] val DefaultTransformerId: TransformerIdentifier =

TransformerIdentifier(ComponentIdentifier.BasedOnParentComponent)

private[core] val TransformerIdSuffix = "Query"

/\*\*

\* For use when building a [[BaseCandidatePipelineQueryTransformer]] in a [[com.twitter.product\_mixer.core.pipeline.PipelineBuilder]]

\* to ensure that the identifier is updated with the parent [[com.twitter.product\_mixer.core.pipeline.Pipeline.identifier]]

\*/

private[core] def copyWithUpdatedIdentifier[Query <: PipelineQuery, CandidateSourceQuery](

queryTransformer: BaseCandidatePipelineQueryTransformer[Query, CandidateSourceQuery],

parentIdentifier: ComponentIdentifier

): BaseCandidatePipelineQueryTransformer[Query, CandidateSourceQuery] = {

if (queryTransformer.identifier == DefaultTransformerId) {

val transformerIdentifierFromParentName = TransformerIdentifier(

s"${parentIdentifier.name}$TransformerIdSuffix")

queryTransformer match {

case queryTransformer: CandidatePipelineQueryTransformer[Query, CandidateSourceQuery] =>

new CandidatePipelineQueryTransformer[Query, CandidateSourceQuery] {

override val identifier: TransformerIdentifier = transformerIdentifierFromParentName

override def transform(input: Query): CandidateSourceQuery =

queryTransformer.transform(input)

}

case queryTransformer: DependentCandidatePipelineQueryTransformer[

Query,

CandidateSourceQuery

] =>

new DependentCandidatePipelineQueryTransformer[Query, CandidateSourceQuery] {

override val identifier: TransformerIdentifier = transformerIdentifierFromParentName

override def transform(

input: Query,

candidates: Seq[CandidateWithDetails]

): CandidateSourceQuery =

queryTransformer.transform(input, candidates)

}

}

} else {

queryTransformer

}

}

}