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

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

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

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

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

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

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

import com.twitter.stitch.Stitch

/\*\*

\* [[CandidateDecorator]] generates a [[com.twitter.product\_mixer.core.model.common.presentation.UniversalPresentation]]

\* for Candidates, which encapsulate information about how to present the candidate

\*

\* @see [[https://docbird.twitter.biz/product-mixer/functional-components.html#candidate-decorator]]

\* @see [[com.twitter.product\_mixer.core.model.common.presentation.UniversalPresentation]]

\*/

trait CandidateDecorator[-Query <: PipelineQuery, -Candidate <: UniversalNoun[Any]]

extends Component {

override val identifier: DecoratorIdentifier = CandidateDecorator.DefaultCandidateDecoratorId

/\*\*

\* Given a Seq of `Candidate`, returns a [[Decoration]] for candidates which should be decorated

\*

\* `Candidate`s which aren't decorated can be omitted from the results

\*/

def apply(

query: Query,

candidates: Seq[CandidateWithFeatures[Candidate]]

): Stitch[Seq[Decoration]]

}

object CandidateDecorator {

private[core] val DefaultCandidateDecoratorId: DecoratorIdentifier =

DecoratorIdentifier(ComponentIdentifier.BasedOnParentComponent)

/\*\*

\* For use when building a [[CandidateDecorator]] 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,

Candidate <: UniversalNoun[Any]

](

decorator: CandidateDecorator[Query, Candidate],

parentIdentifier: ComponentIdentifier

): CandidateDecorator[Query, Candidate] = {

if (decorator.identifier == DefaultCandidateDecoratorId) {

new CandidateDecorator[Query, Candidate] {

override val identifier: DecoratorIdentifier = DecoratorIdentifier(parentIdentifier.name)

override def apply(

query: Query,

candidates: Seq[CandidateWithFeatures[Candidate]]

): Stitch[Seq[Decoration]] = decorator.apply(query, candidates)

}

} else {

decorator

}

}

}