package com.twitter.product\_mixer.component\_library.decorator.urt

import com.twitter.product\_mixer.component\_library.model.presentation.urt.UrtItemPresentation

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

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

import com.twitter.product\_mixer.core.model.marshalling.response.urt.TimelineItem

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

import com.twitter.product\_mixer.core.functional\_component.decorator.CandidateDecorator

import com.twitter.product\_mixer.core.functional\_component.decorator.Decoration

import com.twitter.product\_mixer.core.functional\_component.decorator.urt.builder.CandidateUrtEntryBuilder

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

import com.twitter.stitch.Stitch

/\*\*

\* Decorator that will apply the provided [[CandidateUrtEntryBuilder]] to each candidate independently to make a [[TimelineItem]]

\*/

case class UrtItemCandidateDecorator[

Query <: PipelineQuery,

BuilderInput <: UniversalNoun[Any],

BuilderOutput <: TimelineItem

](

builder: CandidateUrtEntryBuilder[Query, BuilderInput, BuilderOutput],

override val identifier: DecoratorIdentifier = DecoratorIdentifier("UrtItemCandidate"))

extends CandidateDecorator[Query, BuilderInput] {

override def apply(

query: Query,

candidates: Seq[CandidateWithFeatures[BuilderInput]]

): Stitch[Seq[Decoration]] = {

val candidatePresentations = candidates.map { candidate =>

val itemPresentation = UrtItemPresentation(

timelineItem = builder(query, candidate.candidate, candidate.features)

)

Decoration(candidate.candidate, itemPresentation)

}

Stitch.value(candidatePresentations)

}

}