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

import com.twitter.product\_mixer.component\_library.decorator.urt.builder.item.tweet.TweetCandidateUrtItemBuilder

import com.twitter.product\_mixer.component\_library.model.candidate.BaseTweetCandidate

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

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

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.model.common.CandidateWithFeatures

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

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

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

import com.twitter.stitch.Stitch

case class UrtConversationItemCandidateDecorator[

Query <: PipelineQuery,

Candidate <: BaseTweetCandidate

](

tweetCandidateUrtItemBuilder: TweetCandidateUrtItemBuilder[Query, Candidate],

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

extends CandidateDecorator[Query, Candidate] {

override def apply(

query: Query,

candidates: Seq[CandidateWithFeatures[Candidate]]

): Stitch[Seq[Decoration]] = {

val candidatePresentations = candidates.view.zipWithIndex.map {

case (candidate, index) =>

val itemPresentation = new UrtItemPresentation(

timelineItem = tweetCandidateUrtItemBuilder(

pipelineQuery = query,

tweetCandidate = candidate.candidate,

candidateFeatures = candidate.features)

) with ConversationModuleItem {

override val treeDisplay: Option[ModuleItemTreeDisplay] = None

override val dispensable: Boolean = index < candidates.length - 1

}

Decoration(candidate.candidate, itemPresentation)

}

Stitch.value(candidatePresentations)

}

}