package com.twitter.product\_mixer.component\_library.decorator.urt.builder.item.moment

import com.twitter.product\_mixer.component\_library.decorator.urt.builder.item.moment.MomentAnnotationCandidateUrtItemBuilder.MomentAnnotationItemClientEventInfoElement

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

import com.twitter.product\_mixer.core.feature.featuremap.FeatureMap

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

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

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

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

import com.twitter.product\_mixer.core.model.marshalling.response.urt.item.moment.MomentAnnotationItem

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

object MomentAnnotationCandidateUrtItemBuilder {

val MomentAnnotationItemClientEventInfoElement = "metadata"

}

case class MomentAnnotationCandidateUrtItemBuilder[Query <: PipelineQuery](

clientEventInfoBuilder: BaseClientEventInfoBuilder[Query, MomentAnnotationCandidate],

annotationTextRichTextBuilder: BaseRichTextBuilder[Query, MomentAnnotationCandidate],

annotationHeaderRichTextBuilder: BaseRichTextBuilder[Query, MomentAnnotationCandidate],

feedbackActionInfoBuilder: Option[

BaseFeedbackActionInfoBuilder[Query, MomentAnnotationCandidate]

] = None,

) extends CandidateUrtEntryBuilder[Query, MomentAnnotationCandidate, MomentAnnotationItem] {

override def apply(

query: Query,

candidate: MomentAnnotationCandidate,

candidateFeatures: FeatureMap

): MomentAnnotationItem = MomentAnnotationItem(

id = candidate.id,

sortIndex = None, // Sort indexes are automatically set in the domain marshaller phase

clientEventInfo = clientEventInfoBuilder(

query,

candidate,

candidateFeatures,

Some(MomentAnnotationItemClientEventInfoElement)),

feedbackActionInfo =

feedbackActionInfoBuilder.flatMap(\_.apply(query, candidate, candidateFeatures)),

isPinned = None,

text =

candidate.text.map(\_ => annotationTextRichTextBuilder(query, candidate, candidateFeatures)),

header = candidate.header.map(\_ =>

annotationHeaderRichTextBuilder(query, candidate, candidateFeatures)),

)

}