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

import com.twitter.product\_mixer.component\_library.decorator.urt.builder.item.icon\_label.IconLabelCandidateUrtItemBuilder.IconLabelClientEventInfoElement

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

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.icon.HorizonIcon

import com.twitter.product\_mixer.core.model.marshalling.response.urt.item.icon\_label.IconLabelItem

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

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

object IconLabelCandidateUrtItemBuilder {

val IconLabelClientEventInfoElement: String = "iconlabel"

}

case class IconLabelCandidateUrtItemBuilder[-Query <: PipelineQuery, Candidate <: LabelCandidate](

richTextBuilder: BaseRichTextBuilder[Query, Candidate],

icon: Option[HorizonIcon] = None,

entities: Option[List[RichTextEntity]] = None,

clientEventInfoBuilder: Option[BaseClientEventInfoBuilder[Query, Candidate]] = None,

feedbackActionInfoBuilder: Option[BaseFeedbackActionInfoBuilder[Query, Candidate]] = None)

extends CandidateUrtEntryBuilder[Query, Candidate, IconLabelItem] {

override def apply(

query: Query,

labelCandidate: Candidate,

candidateFeatures: FeatureMap

): IconLabelItem =

IconLabelItem(

id = labelCandidate.id.toString,

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

clientEventInfo = clientEventInfoBuilder.flatMap(

\_.apply(query, labelCandidate, candidateFeatures, Some(IconLabelClientEventInfoElement))),

feedbackActionInfo =

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

text = richTextBuilder(query, labelCandidate, candidateFeatures),

icon = icon,

)

}