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

import com.twitter.product\_mixer.component\_library.decorator.urt.builder.item.message.InlinePromptCandidateUrtItemStringCenterBuilder.InlinePromptClientEventInfoElement

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

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.metadata.BaseStr

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

import com.twitter.product\_mixer.core.functional\_component.decorator.urt.builder.social\_context.BaseSocialContextBuilder

import com.twitter.product\_mixer.core.model.marshalling.response.urt.item.message.InlinePromptMessageContent

import com.twitter.product\_mixer.core.model.marshalling.response.urt.item.message.MessagePromptItem

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

object InlinePromptCandidateUrtItemStringCenterBuilder {

val InlinePromptClientEventInfoElement: String = "message"

}

case class InlinePromptCandidateUrtItemStringCenterBuilder[-Query <: PipelineQuery](

clientEventInfoBuilder: BaseClientEventInfoBuilder[Query, InlinePromptCandidate],

feedbackActionInfoBuilder: Option[

BaseFeedbackActionInfoBuilder[Query, InlinePromptCandidate]

] = None,

headerTextBuilder: BaseStr[Query, InlinePromptCandidate],

bodyTextBuilder: Option[BaseStr[Query, InlinePromptCandidate]] = None,

headerRichTextBuilder: Option[BaseRichTextBuilder[Query, InlinePromptCandidate]] = None,

bodyRichTextBuilder: Option[BaseRichTextBuilder[Query, InlinePromptCandidate]] = None,

primaryMessageTextActionBuilder: Option[

MessageTextActionBuilder[Query, InlinePromptCandidate]

] = None,

secondaryMessageTextActionBuilder: Option[

MessageTextActionBuilder[Query, InlinePromptCandidate]

] = None,

socialContextBuilder: Option[BaseSocialContextBuilder[Query, InlinePromptCandidate]] = None,

userFacePileBuilder: Option[

UserFacePileBuilder

] = None)

extends CandidateUrtEntryBuilder[Query, InlinePromptCandidate, MessagePromptItem] {

override def apply(

query: Query,

inlinePromptCandidate: InlinePromptCandidate,

candidateFeatures: FeatureMap

): MessagePromptItem =

MessagePromptItem(

id = inlinePromptCandidate.id,

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

clientEventInfo = clientEventInfoBuilder(

query,

inlinePromptCandidate,

candidateFeatures,

Some(InlinePromptClientEventInfoElement)),

feedbackActionInfo =

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

isPinned = None,

content = InlinePromptMessageContent(

headerText = headerTextBuilder.apply(query, inlinePromptCandidate, candidateFeatures),

bodyText = bodyTextBuilder.map(\_.apply(query, inlinePromptCandidate, candidateFeatures)),

primaryButtonAction = primaryMessageTextActionBuilder.map(

\_.apply(query, inlinePromptCandidate, candidateFeatures)),

secondaryButtonAction = secondaryMessageTextActionBuilder.map(

\_.apply(query, inlinePromptCandidate, candidateFeatures)),

headerRichText =

headerRichTextBuilder.map(\_.apply(query, inlinePromptCandidate, candidateFeatures)),

bodyRichText =

bodyRichTextBuilder.map(\_.apply(query, inlinePromptCandidate, candidateFeatures)),

socialContext =

socialContextBuilder.flatMap(\_.apply(query, inlinePromptCandidate, candidateFeatures)),

userFacepile = userFacePileBuilder.map(\_.apply())

),

impressionCallbacks = None

)

}