package com.twitter.product\_mixer.component\_library.decorator.urt.builder.flexible\_injection\_pipeline

import com.twitter.onboarding.injections.thriftscala.Injection

import com.twitter.onboarding.injections.{thriftscala => onboardingthrift}

import com.twitter.product\_mixer.component\_library.decorator.urt.builder.flexible\_injection\_pipeline.OnboardingInjectionConversions.\_

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

import com.twitter.product\_mixer.component\_library.pipeline.candidate.flexible\_injection\_pipeline.transformer.FlipPromptCarouselTileFeature

import com.twitter.product\_mixer.component\_library.pipeline.candidate.flexible\_injection\_pipeline.transformer.FlipPromptInjectionsFeature

import com.twitter.product\_mixer.component\_library.pipeline.candidate.flexible\_injection\_pipeline.transformer.FlipPromptOffsetInModuleFeature

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.marshaller.TransportMarshaller

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

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

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

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

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

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

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

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

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.MessageContent

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

import com.twitter.product\_mixer.core.model.marshalling.response.urt.item.prompt.PromptItem

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

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

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

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

object FlipPromptCandidateUrtItemBuilder {

val FlipPromptClientEventInfoElement: String = "flip-prompt-message"

}

case class FlipPromptCandidateUrtItemBuilder[-Query <: PipelineQuery]()

extends CandidateUrtEntryBuilder[Query, BasePromptCandidate[Any], TimelineItem] {

override def apply(

query: Query,

promptCandidate: BasePromptCandidate[Any],

candidateFeatures: FeatureMap

): TimelineItem = {

val injection = candidateFeatures.get(FlipPromptInjectionsFeature)

injection match {

case onboardingthrift.Injection.InlinePrompt(candidate) =>

MessagePromptItem(

id = promptCandidate.id.toString,

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

clientEventInfo = buildClientEventInfo(injection),

feedbackActionInfo = candidate.feedbackInfo.map(convertFeedbackInfo),

isPinned = Some(candidate.isPinnedEntry),

content = getInlinePromptMessageContent(candidate),

impressionCallbacks = candidate.impressionCallbacks.map(\_.map(convertCallback).toList)

)

case onboardingthrift.Injection.FullCover(candidate) =>

FullCover(

id = promptCandidate.id.toString,

// Note that sort index is not used for Covers, as they are not TimelineEntry and do not have entryId

sortIndex = None,

clientEventInfo =

Some(OnboardingInjectionConversions.convertClientEventInfo(candidate.clientEventInfo)),

content = getFullCoverContent(candidate)

)

case onboardingthrift.Injection.HalfCover(candidate) =>

HalfCover(

id = promptCandidate.id.toString,

// Note that sort index is not used for Covers, as they are not TimelineEntry and do not have entryId

sortIndex = None,

clientEventInfo =

Some(OnboardingInjectionConversions.convertClientEventInfo(candidate.clientEventInfo)),

content = getHalfCoverContent(candidate)

)

case Injection.TilesCarousel(\_) =>

val offsetInModuleOption =

candidateFeatures.get(FlipPromptOffsetInModuleFeature)

val offsetInModule =

offsetInModuleOption.getOrElse(throw FlipPromptOffsetInModuleMissing)

val tileOption =

candidateFeatures.get(FlipPromptCarouselTileFeature)

val tile = tileOption.getOrElse(throw FlipPromptCarouselTileMissing)

TilesCarouselConversions.convertTile(tile, offsetInModule)

case onboardingthrift.Injection.RelevancePrompt(candidate) =>

PromptItem(

id = promptCandidate.id.toString,

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

clientEventInfo = buildClientEventInfo(injection),

content = RelevancePromptConversions.convertContent(candidate),

impressionCallbacks = Some(candidate.impressionCallbacks.map(convertCallback).toList)

)

case \_ => throw new UnsupportedFlipPromptException(injection)

}

}

private def getInlinePromptMessageContent(

candidate: onboardingthrift.InlinePrompt

): MessageContent = {

candidate.image match {

case Some(image) =>

HeaderImagePromptMessageContent(

headerImage = convertImage(image),

headerText = Some(candidate.headerText.text),

bodyText = candidate.bodyText.map(\_.text),

primaryButtonAction = candidate.primaryAction.map(convertButtonAction),

secondaryButtonAction = candidate.secondaryAction.map(convertButtonAction),

headerRichText = Some(convertRichText(candidate.headerText)),

bodyRichText = candidate.bodyText.map(convertRichText),

action =

None

)

case None =>

InlinePromptMessageContent(

headerText = candidate.headerText.text,

bodyText = candidate.bodyText.map(\_.text),

primaryButtonAction = candidate.primaryAction.map(convertButtonAction),

secondaryButtonAction = candidate.secondaryAction.map(convertButtonAction),

headerRichText = Some(convertRichText(candidate.headerText)),

bodyRichText = candidate.bodyText.map(convertRichText),

socialContext = candidate.socialContext.map(convertSocialContext),

userFacepile = candidate.promptUserFacepile.map(convertUserFacePile)

)

}

}

private def getFullCoverContent(

candidate: onboardingthrift.FullCover

): FullCoverContent =

FullCoverContent(

displayType = CoverFullCoverDisplayType,

primaryText = convertRichText(candidate.primaryText),

primaryCoverCta = convertCoverCta(candidate.primaryButtonAction),

secondaryCoverCta = candidate.secondaryButtonAction.map(convertCoverCta),

secondaryText = candidate.secondaryText.map(convertRichText),

imageVariant = candidate.image.map(img => convertImageVariant(img.image)),

details = candidate.detailText.map(convertRichText),

dismissInfo = candidate.dismissInfo.map(convertDismissInfo),

imageDisplayType = candidate.image.map(img => convertImageDisplayType(img.imageDisplayType)),

impressionCallbacks = candidate.impressionCallbacks.map(\_.map(convertCallback).toList)

)

private def getHalfCoverContent(

candidate: onboardingthrift.HalfCover

): HalfCoverContent =

HalfCoverContent(

displayType =

candidate.displayType.map(convertHalfCoverDisplayType).getOrElse(CoverHalfCoverDisplayType),

primaryText = convertRichText(candidate.primaryText),

primaryCoverCta = convertCoverCta(candidate.primaryButtonAction),

secondaryCoverCta = candidate.secondaryButtonAction.map(convertCoverCta),

secondaryText = candidate.secondaryText.map(convertRichText),

coverImage = candidate.image.map(convertCoverImage),

dismissible = candidate.dismissible,

dismissInfo = candidate.dismissInfo.map(convertDismissInfo),

impressionCallbacks = candidate.impressionCallbacks.map(\_.map(convertCallback).toList)

)

private def buildClientEventInfo(

injection: Injection

): Option[ClientEventInfo] = {

injection match {

//To keep parity between TimelineMixer and Product Mixer, inline prompt switches sets the prompt product identifier as the component and no element. Also includes clientEventDetails

case onboardingthrift.Injection.InlinePrompt(candidate) =>

val clientEventDetails: ClientEventDetails =

ClientEventDetails(

conversationDetails = None,

timelinesDetails = Some(TimelinesDetails(injectionType = Some("Message"), None, None)),

articleDetails = None,

liveEventDetails = None,

commerceDetails = None

)

Some(

ClientEventInfo(

component = candidate.injectionIdentifier,

element = None,

details = Some(clientEventDetails),

action = None,

entityToken = None))

// To keep parity between TLM and PM we swap component and elements.

case onboardingthrift.Injection.RelevancePrompt(candidate) =>

Some(

ClientEventInfo(

// Identifier is prefixed with onboarding per TLM

component = Some("onboarding\_" + candidate.injectionIdentifier),

element = Some("relevance\_prompt"),

details = None,

action = None,

entityToken = None

))

case \_ => None

}

}

}

class UnsupportedFlipPromptException(injection: onboardingthrift.Injection)

extends UnsupportedOperationException(

"Unsupported timeline item " + TransportMarshaller.getSimpleName(injection.getClass))

object FlipPromptOffsetInModuleMissing

extends NoSuchElementException(

"FlipPromptOffsetInModuleFeature must be set for the TilesCarousel FLIP injection in PromptCandidateSource")

object FlipPromptCarouselTileMissing

extends NoSuchElementException(

"FlipPromptCarouselTileFeature must be set for the TilesCarousel FLIP injection in PromptCandidateSource")