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

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

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

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

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

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

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

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

/\*\*\*

\* Helper class to convert Relevance Prompt related onboarding thrift to product-mixer models

\*/

object RelevancePromptConversions {

def convertContent(

candidate: onboardingthrift.RelevancePrompt

): RelevancePromptContent =

RelevancePromptContent(

displayType = convertDisplayType(candidate.displayType),

title = candidate.title.text,

confirmation = buildConfirmation(candidate),

isRelevantText = candidate.isRelevantButton.text,

notRelevantText = candidate.notRelevantButton.text,

isRelevantCallback = convertCallbacks(candidate.isRelevantButton.callbacks),

notRelevantCallback = convertCallbacks(candidate.notRelevantButton.callbacks),

isRelevantFollowUp = None,

notRelevantFollowUp = None

)

// Based on com.twitter.timelinemixer.injection.model.candidate.OnboardingRelevancePromptDisplayType#fromThrift

def convertDisplayType(

displayType: onboardingthrift.RelevancePromptDisplayType

): RelevancePromptDisplayType =

displayType match {

case onboardingthrift.RelevancePromptDisplayType.Normal => Normal

case onboardingthrift.RelevancePromptDisplayType.Compact => Compact

case onboardingthrift.RelevancePromptDisplayType.Large => Large

case onboardingthrift.RelevancePromptDisplayType

.EnumUnknownRelevancePromptDisplayType(value) =>

throw new UnsupportedOperationException(s"Unknown display type: $value")

}

// Based on com.twitter.timelinemixer.injection.model.injection.OnboardingRelevancePromptInjection#buildConfirmation

def buildConfirmation(candidate: onboardingthrift.RelevancePrompt): String = {

val isRelevantTextConfirmation =

buttonToDismissFeedbackText(candidate.isRelevantButton).getOrElse("")

val notRelevantTextConfirmation =

buttonToDismissFeedbackText(candidate.notRelevantButton).getOrElse("")

if (isRelevantTextConfirmation != notRelevantTextConfirmation)

throw new IllegalArgumentException(

s"""confirmation text not consistent for two buttons, :

isRelevantConfirmation: ${isRelevantTextConfirmation}

notRelevantConfirmation: ${notRelevantTextConfirmation}

"""

)

isRelevantTextConfirmation

}

// Based on com.twitter.timelinemixer.injection.model.candidate.OnboardingInjectionAction#fromThrift

def buttonToDismissFeedbackText(button: onboardingthrift.ButtonAction): Option[String] = {

button.buttonBehavior match {

case onboardingthrift.ButtonBehavior.Dismiss(d) => d.feedbackMessage.map(\_.text)

case \_ => None

}

}

// Based on com.twitter.timelinemixer.injection.model.injection.OnboardingRelevancePromptInjection#buildCallback

def convertCallbacks(onboardingCallbacks: Option[Seq[onboardingthrift.Callback]]): Callback = {

OnboardingInjectionConversions.convertCallback(

onboardingCallbacks

.flatMap(\_.headOption)

.getOrElse(

throw new NoSuchElementException(s"Callback must be provided for the Relevance Prompt")

))

}

}