package com.twitter.frigate.pushservice.predicate.ntab\_caret\_fatigue

import com.twitter.notificationservice.thriftscala.GenericType

import com.twitter.frigate.thriftscala.FrigateNotification

import com.twitter.notificationservice.genericfeedbackstore.FeedbackPromptValue

import com.twitter.notificationservice.thriftscala.CaretFeedbackDetails

import com.twitter.notificationservice.feedback.thriftscala.FeedbackMetadata

import com.twitter.notificationservice.feedback.thriftscala.InlineFeedback

import com.twitter.notificationservice.feedback.thriftscala.FeedbackValue

import com.twitter.notificationservice.feedback.thriftscala.YesOrNoAnswer

object FeedbackTypeEnum extends Enumeration {

val Unknown = Value

val CaretDislike = Value

val InlineDislike = Value

val InlineLike = Value

val InlineRevertedLike = Value

val InlineRevertedDislike = Value

val PromptRelevant = Value

val PromptIrrelevant = Value

val InlineDismiss = Value

val InlineRevertedDismiss = Value

val InlineSeeLess = Value

val InlineRevertedSeeLess = Value

val InlineNotRelevant = Value

val InlineRevertedNotRelevant = Value

def safeFindByName(name: String): Value =

values.find(\_.toString.toLowerCase() == name.toLowerCase()).getOrElse(Unknown)

}

trait FeedbackModel {

def timestampMs: Long

def feedbackTypeEnum: FeedbackTypeEnum.Value

def notificationImpressionId: Option[String]

def notification: Option[FrigateNotification] = None

}

case class CaretFeedbackModel(

caretFeedbackDetails: CaretFeedbackDetails,

notificationOpt: Option[FrigateNotification] = None)

extends FeedbackModel {

override def timestampMs: Long = caretFeedbackDetails.eventTimestamp

override def feedbackTypeEnum: FeedbackTypeEnum.Value = FeedbackTypeEnum.CaretDislike

override def notificationImpressionId: Option[String] = caretFeedbackDetails.impressionId

override def notification: Option[FrigateNotification] = notificationOpt

def notificationGenericType: Option[GenericType] = {

caretFeedbackDetails.genericNotificationMetadata match {

case Some(genericNotificationMetadata) =>

Some(genericNotificationMetadata.genericType)

case None => None

}

}

}

case class InlineFeedbackModel(

feedback: FeedbackPromptValue,

notificationOpt: Option[FrigateNotification] = None)

extends FeedbackModel {

override def timestampMs: Long = feedback.createdAt.inMilliseconds

override def feedbackTypeEnum: FeedbackTypeEnum.Value = {

feedback.feedbackValue match {

case FeedbackValue(

\_,

\_,

\_,

Some(FeedbackMetadata.InlineFeedback(InlineFeedback(Some(answer))))) =>

FeedbackTypeEnum.safeFindByName("inline" + answer)

case \_ => FeedbackTypeEnum.Unknown

}

}

override def notificationImpressionId: Option[String] = Some(feedback.feedbackValue.impressionId)

override def notification: Option[FrigateNotification] = notificationOpt

}

case class PromptFeedbackModel(

feedback: FeedbackPromptValue,

notificationOpt: Option[FrigateNotification] = None)

extends FeedbackModel {

override def timestampMs: Long = feedback.createdAt.inMilliseconds

override def feedbackTypeEnum: FeedbackTypeEnum.Value = {

feedback.feedbackValue match {

case FeedbackValue(\_, \_, \_, Some(FeedbackMetadata.YesOrNoAnswer(answer))) =>

answer match {

case YesOrNoAnswer.Yes => FeedbackTypeEnum.PromptRelevant

case YesOrNoAnswer.No => FeedbackTypeEnum.PromptIrrelevant

case \_ => FeedbackTypeEnum.Unknown

}

case \_ => FeedbackTypeEnum.Unknown

}

}

override def notificationImpressionId: Option[String] = Some(feedback.feedbackValue.impressionId)

override def notification: Option[FrigateNotification] = notificationOpt

}

object FeedbackModelHydrator {

def HydrateNotification(

feedbacks: Seq[FeedbackModel],

history: Seq[FrigateNotification]

): Seq[FeedbackModel] = {

feedbacks.map {

case feedback @ (inlineFeedback: InlineFeedbackModel) =>

inlineFeedback.copy(notificationOpt = history.find(

\_.impressionId

.equals(feedback.notificationImpressionId)))

case feedback @ (caretFeedback: CaretFeedbackModel) =>

caretFeedback.copy(notificationOpt = history.find(

\_.impressionId

.equals(feedback.notificationImpressionId)))

case feedback @ (promptFeedback: PromptFeedbackModel) =>

promptFeedback.copy(notificationOpt = history.find(

\_.impressionId

.equals(feedback.notificationImpressionId)))

case feedback => feedback

}

}

}