package com.twitter.frigate.pushservice.model.ntab

import com.twitter.frigate.common.rec\_types.RecTypes

import com.twitter.frigate.pushservice.model.PushTypes.PushCandidate

import com.twitter.frigate.pushservice.params.PushConstants

import com.twitter.frigate.thriftscala.{CommonRecommendationType => CRT}

import com.twitter.notificationservice.thriftscala.\_

import com.twitter.util.Future

import com.twitter.util.Time

trait DiscoverTwitterNtabRequestHydrator extends NTabRequestHydrator {

self: PushCandidate =>

override val senderIdFut: Future[Long] = Future.value(0L)

override val tapThroughFut: Future[String] =

commonRecType match {

case CRT.AddressBookUploadPush => Future.value(PushConstants.AddressBookUploadTapThrough)

case CRT.InterestPickerPush => Future.value(PushConstants.InterestPickerTapThrough)

case CRT.CompleteOnboardingPush =>

Future.value(PushConstants.CompleteOnboardingInterestAddressTapThrough)

case \_ =>

Future.value(PushConstants.ConnectTabPushTapThrough)

}

override val displayTextEntitiesFut: Future[Seq[DisplayTextEntity]] = Future.Nil

override val facepileUsersFut: Future[Seq[Long]] = Future.Nil

override val storyContext: Option[StoryContext] = None

override val inlineCard: Option[InlineCard] = None

override val socialProofDisplayText: Option[DisplayText] = Some(DisplayText())

override lazy val ntabRequest: Future[Option[CreateGenericNotificationRequest]] =

if (self.commonRecType == CRT.ConnectTabPush || RecTypes.isOnboardingFlowType(

self.commonRecType)) {

Future.join(senderIdFut, displayTextEntitiesFut, facepileUsersFut, tapThroughFut).map {

case (senderId, displayTextEntities, facepileUsers, tapThrough) =>

Some(

CreateGenericNotificationRequest(

userId = target.targetId,

senderId = senderId,

genericType = GenericType.RefreshableNotification,

displayText = DisplayText(values = displayTextEntities),

facepileUsers = facepileUsers,

timestampMillis = Time.now.inMillis,

tapThroughAction = Some(TapThroughAction(Some(tapThrough))),

impressionId = Some(impressionId),

socialProofText = socialProofDisplayText,

context = storyContext,

inlineCard = inlineCard,

refreshableType = refreshableType

))

}

} else Future.None

}