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

import com.twitter.frigate.pushservice.model.TopicProofTweetPushCandidate

import com.twitter.frigate.pushservice.exception.TweetNTabRequestHydratorException

import com.twitter.frigate.pushservice.exception.UttEntityNotFoundException

import com.twitter.frigate.pushservice.take.NotificationServiceSender

import com.twitter.notificationservice.thriftscala.DisplayText

import com.twitter.notificationservice.thriftscala.DisplayTextEntity

import com.twitter.notificationservice.thriftscala.StoryContext

import com.twitter.notificationservice.thriftscala.StoryContextValue

import com.twitter.notificationservice.thriftscala.TextValue

import com.twitter.util.Future

trait TopicProofTweetNtabRequestHydrator extends NTabRequestHydrator {

self: TopicProofTweetPushCandidate =>

override def displayTextEntitiesFut: Future[Seq[DisplayTextEntity]] = NotificationServiceSender

.getDisplayTextEntityFromUser(tweetAuthor, "tweetAuthorName", true)

.map(\_.toSeq)

private lazy val uttEntity = localizedUttEntity.getOrElse(

throw new UttEntityNotFoundException(

s"${getClass.getSimpleName} UttEntity missing for $tweetId")

)

override lazy val tapThroughFut: Future[String] = {

tweetAuthor.map {

case Some(author) =>

val authorProfile = author.profile.getOrElse(

throw new TweetNTabRequestHydratorException(

s"Unable to obtain author profile for: ${author.id}"))

s"${authorProfile.screenName}/status/${tweetId.toString}"

case \_ =>

throw new TweetNTabRequestHydratorException(

s"Unable to obtain author and target details to generate tap through for Tweet: $tweetId")

}

}

override lazy val socialProofDisplayText: Option[DisplayText] = {

Some(

DisplayText(values =

Seq(DisplayTextEntity("topic\_name", TextValue.Text(uttEntity.localizedNameForDisplay))))

)

}

override lazy val facepileUsersFut: Future[Seq[Long]] = senderIdFut.map(Seq(\_))

override val inlineCard = None

override def storyContext: Option[StoryContext] = Some(

StoryContext("", Some(StoryContextValue.Tweets(Seq(tweetId)))))

override def senderIdFut: Future[Long] =

tweetAuthor.map {

case Some(author) => author.id

case \_ =>

throw new TweetNTabRequestHydratorException(

s"Unable to obtain Author ID for: $commonRecType")

}

}