package com.twitter.frigate.pushservice.util

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.frigate.common.base.\_

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

import com.twitter.frigate.thriftscala.FrigateNotification

import com.twitter.frigate.thriftscala.NotificationDisplayLocation

object Candidate2FrigateNotification {

def getFrigateNotification(

candidate: PushCandidate

)(

implicit statsReceiver: StatsReceiver

): FrigateNotification = {

candidate match {

case topicTweetCandidate: PushCandidate with BaseTopicTweetCandidate =>

PushAdaptorUtil.getFrigateNotificationForTweet(

crt = topicTweetCandidate.commonRecType,

tweetId = topicTweetCandidate.tweetId,

scActions = Nil,

authorIdOpt = topicTweetCandidate.authorId,

pushCopyId = topicTweetCandidate.pushCopyId,

ntabCopyId = topicTweetCandidate.ntabCopyId,

simclusterId = None,

semanticCoreEntityIds = topicTweetCandidate.semanticCoreEntityId.map(List(\_)),

candidateContent = topicTweetCandidate.content,

trendId = None

)

case trendTweetCandidate: PushCandidate with TrendTweetCandidate =>

PushAdaptorUtil.getFrigateNotificationForTweet(

trendTweetCandidate.commonRecType,

trendTweetCandidate.tweetId,

Nil,

trendTweetCandidate.authorId,

trendTweetCandidate.pushCopyId,

trendTweetCandidate.ntabCopyId,

None,

None,

trendTweetCandidate.content,

Some(trendTweetCandidate.trendId)

)

case tripTweetCandidate: PushCandidate with OutOfNetworkTweetCandidate with TripCandidate =>

PushAdaptorUtil.getFrigateNotificationForTweet(

crt = tripTweetCandidate.commonRecType,

tweetId = tripTweetCandidate.tweetId,

scActions = Nil,

authorIdOpt = tripTweetCandidate.authorId,

pushCopyId = tripTweetCandidate.pushCopyId,

ntabCopyId = tripTweetCandidate.ntabCopyId,

simclusterId = None,

semanticCoreEntityIds = None,

candidateContent = tripTweetCandidate.content,

trendId = None,

tweetTripDomain = tripTweetCandidate.tripDomain

)

case outOfNetworkTweetCandidate: PushCandidate with OutOfNetworkTweetCandidate =>

PushAdaptorUtil.getFrigateNotificationForTweet(

crt = outOfNetworkTweetCandidate.commonRecType,

tweetId = outOfNetworkTweetCandidate.tweetId,

scActions = Nil,

authorIdOpt = outOfNetworkTweetCandidate.authorId,

pushCopyId = outOfNetworkTweetCandidate.pushCopyId,

ntabCopyId = outOfNetworkTweetCandidate.ntabCopyId,

simclusterId = None,

semanticCoreEntityIds = None,

candidateContent = outOfNetworkTweetCandidate.content,

trendId = None

)

case userCandidate: PushCandidate with UserCandidate with SocialContextActions =>

PushAdaptorUtil.getFrigateNotificationForUser(

userCandidate.commonRecType,

userCandidate.userId,

userCandidate.socialContextActions,

userCandidate.pushCopyId,

userCandidate.ntabCopyId

)

case userCandidate: PushCandidate with UserCandidate =>

PushAdaptorUtil.getFrigateNotificationForUser(

userCandidate.commonRecType,

userCandidate.userId,

Nil,

userCandidate.pushCopyId,

userCandidate.ntabCopyId

)

case tweetCandidate: PushCandidate with TweetCandidate with TweetDetails with SocialContextActions =>

PushAdaptorUtil.getFrigateNotificationForTweetWithSocialContextActions(

tweetCandidate.commonRecType,

tweetCandidate.tweetId,

tweetCandidate.socialContextActions,

tweetCandidate.authorId,

tweetCandidate.pushCopyId,

tweetCandidate.ntabCopyId,

candidateContent = tweetCandidate.content,

semanticCoreEntityIds = None,

trendId = None

)

case pushCandidate: PushCandidate =>

FrigateNotification(

commonRecommendationType = pushCandidate.commonRecType,

notificationDisplayLocation = NotificationDisplayLocation.PushToMobileDevice,

pushCopyId = pushCandidate.pushCopyId,

ntabCopyId = pushCandidate.ntabCopyId

)

case \_ =>

statsReceiver

.scope(s"${candidate.commonRecType}").counter("frigate\_notification\_error").incr()

throw new IllegalStateException("Incorrect candidate type when create FrigateNotification")

}

}

}