package com.twitter.frigate.pushservice.model.ibis

import com.twitter.frigate.common.base.OutOfNetworkTweetCandidate

import com.twitter.frigate.common.base.TopicCandidate

import com.twitter.frigate.common.base.TweetAuthorDetails

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

import com.twitter.frigate.common.util.MrPushCopyObjects

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

import com.twitter.frigate.pushservice.params.PushFeatureSwitchParams

import com.twitter.frigate.pushservice.util.InlineActionUtil

import com.twitter.frigate.pushservice.util.PushIbisUtil.mergeModelValues

import com.twitter.frigate.thriftscala.CommonRecommendationType

import com.twitter.util.Future

trait OutOfNetworkTweetIbis2HydratorForCandidate extends TweetCandidateIbis2Hydrator {

self: PushCandidate with OutOfNetworkTweetCandidate with TopicCandidate with TweetAuthorDetails =>

private lazy val useNewOonCopyValue =

if (target.params(PushFeatureSwitchParams.EnableNewMROONCopyForPush)) {

Map(

"use\_new\_oon\_copy" -> "true"

)

} else Map.empty[String, String]

override lazy val tweetDynamicInlineActionsModelValues =

if (target.params(PushFeatureSwitchParams.EnableOONGeneratedInlineActions)) {

val actions = target.params(PushFeatureSwitchParams.OONTweetDynamicInlineActionsList)

InlineActionUtil.getGeneratedTweetInlineActions(target, statsReceiver, actions)

} else Map.empty[String, String]

private lazy val ibtModelValues: Map[String, String] =

Map(

"is\_tweet" -> s"${!(hasPhoto || hasVideo)}",

"is\_photo" -> s"$hasPhoto",

"is\_video" -> s"$hasVideo"

)

private lazy val launchVideosInImmersiveExploreValue =

Map(

"launch\_videos\_in\_immersive\_explore" -> s"${hasVideo && target.params(PushFeatureSwitchParams.EnableLaunchVideosInImmersiveExplore)}"

)

private lazy val oonTweetModelValues =

useNewOonCopyValue ++ ibtModelValues ++ tweetDynamicInlineActionsModelValues ++ launchVideosInImmersiveExploreValue

lazy val useTopicCopyForMBCGIbis = mrModelingBasedTypes.contains(commonRecType) && target.params(

PushFeatureSwitchParams.EnableMrModelingBasedCandidatesTopicCopy)

lazy val useTopicCopyForFrsIbis = frsTypes.contains(commonRecType) && target.params(

PushFeatureSwitchParams.EnableFrsTweetCandidatesTopicCopy)

lazy val useTopicCopyForTagspaceIbis = tagspaceTypes.contains(commonRecType) && target.params(

PushFeatureSwitchParams.EnableHashspaceCandidatesTopicCopy)

override lazy val modelName: String = {

if (localizedUttEntity.isDefined &&

(useTopicCopyForMBCGIbis || useTopicCopyForFrsIbis || useTopicCopyForTagspaceIbis)) {

MrPushCopyObjects.TopicTweet.ibisPushModelName // uses topic copy

} else super.modelName

}

lazy val exploreVideoParams: Map[String, String] = {

if (self.commonRecType == CommonRecommendationType.ExploreVideoTweet) {

Map(

"is\_explore\_video" -> "true"

)

} else Map.empty[String, String]

}

override lazy val customFieldsMapFut: Future[Map[String, String]] =

mergeModelValues(super.customFieldsMapFut, exploreVideoParams)

override lazy val tweetModelValues: Future[Map[String, String]] =

if (localizedUttEntity.isDefined &&

(useTopicCopyForMBCGIbis || useTopicCopyForFrsIbis || useTopicCopyForTagspaceIbis)) {

lazy val topicTweetModelValues: Map[String, String] =

Map("topic\_name" -> s"${localizedUttEntity.get.localizedNameForDisplay}")

for {

superModelValues <- super.tweetModelValues

tweetInlineModelValue <- tweetInlineActionModelValue

} yield {

superModelValues ++ topicTweetModelValues ++ tweetInlineModelValue

}

} else {

for {

superModelValues <- super.tweetModelValues

tweetInlineModelValues <- tweetInlineActionModelValue

} yield {

superModelValues ++ mediaModelValue ++ oonTweetModelValues ++ tweetInlineModelValues ++ inlineVideoMediaMap

}

}

}