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

import com.twitter.frigate.pushservice.model.SubscribedSearchTweetPushCandidate

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

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

import com.twitter.util.Future

trait SubscribedSearchTweetIbis2Hydrator extends TweetCandidateIbis2Hydrator {

self: SubscribedSearchTweetPushCandidate =>

override lazy val tweetDynamicInlineActionsModelValues = {

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

val actions = target.params(PushFeatureSwitchParams.TweetDynamicInlineActionsList)

InlineActionUtil.getGeneratedTweetInlineActions(target, statsReceiver, actions)

} else Map.empty[String, String]

}

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

Map(

"search\_term" -> searchTerm,

"search\_url" -> pushLandingUrl

)

private lazy val searchModelValues = searchTermValue ++ tweetDynamicInlineActionsModelValues

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

for {

superModelValues <- super.tweetModelValues

tweetInlineModelValues <- tweetInlineActionModelValue

} yield {

superModelValues ++ mediaModelValue ++ searchModelValues ++ tweetInlineModelValues ++ inlineVideoMediaMap

}

}