package com.twitter.home\_mixer.product.for\_you.feature\_hydrator

import com.twitter.home\_mixer.model.HomeFeatures.AuthorIdFeature

import com.twitter.home\_mixer.model.HomeFeatures.IsHydratedFeature

import com.twitter.home\_mixer.model.HomeFeatures.TweetTextFeature

import com.twitter.product\_mixer.component\_library.model.candidate.BaseTweetCandidate

import com.twitter.product\_mixer.core.feature.Feature

import com.twitter.product\_mixer.core.feature.featuremap.FeatureMap

import com.twitter.product\_mixer.core.feature.featuremap.FeatureMapBuilder

import com.twitter.product\_mixer.core.functional\_component.feature\_hydrator.CandidateFeatureHydrator

import com.twitter.product\_mixer.core.model.common.identifier.FeatureHydratorIdentifier

import com.twitter.product\_mixer.core.pipeline.PipelineQuery

import com.twitter.spam.rtf.{thriftscala => rtf}

import com.twitter.stitch.Stitch

import com.twitter.stitch.tweetypie.{TweetyPie => TweetypieStitchClient}

import com.twitter.tweetypie.{thriftscala => TP}

import javax.inject.Inject

import javax.inject.Singleton

@Singleton

class TweetPreviewTweetypieCandidateFeatureHydrator @Inject() (

tweetypieStitchClient: TweetypieStitchClient)

extends CandidateFeatureHydrator[PipelineQuery, BaseTweetCandidate] {

private val CoreTweetFields: Set[TP.TweetInclude] = Set[TP.TweetInclude](

TP.TweetInclude.TweetFieldId(TP.Tweet.IdField.id),

TP.TweetInclude.TweetFieldId(TP.Tweet.CoreDataField.id)

)

private val DefaultFeatureMap = FeatureMapBuilder()

.add(TweetTextFeature, None)

.add(IsHydratedFeature, false)

.add(AuthorIdFeature, None)

.build()

override val features: Set[Feature[\_, \_]] =

Set(TweetTextFeature, IsHydratedFeature, AuthorIdFeature)

override val identifier: FeatureHydratorIdentifier =

FeatureHydratorIdentifier("TweetPreviewTweetypie")

override def apply(

query: PipelineQuery,

candidate: BaseTweetCandidate,

existingFeatures: FeatureMap

): Stitch[FeatureMap] = {

tweetypieStitchClient

.getTweetFields(

tweetId = candidate.id,

options = TP.GetTweetFieldsOptions(

tweetIncludes = CoreTweetFields,

includeRetweetedTweet = false,

includeQuotedTweet = false,

visibilityPolicy = TP.TweetVisibilityPolicy.UserVisible,

safetyLevel = Some(rtf.SafetyLevel.TimelineHomeTweetPreview),

forUserId = query.getOptionalUserId

)

).map {

case TP.GetTweetFieldsResult(\_, TP.TweetFieldsResultState.Found(found), quoteOpt, \_) =>

val tweetText = found.tweet.coreData.map(\_.text)

FeatureMapBuilder()

.add(TweetTextFeature, tweetText)

.add(IsHydratedFeature, true)

.add(AuthorIdFeature, found.tweet.coreData.map(\_.userId))

.build()

// If no tweet result found, return default features

case \_ =>

DefaultFeatureMap

}

}

}