package com.twitter.tweetypie.util

import com.twitter.tweetypie.thriftscala.\_

object TweetTransformer {

def toStatus(tweet: Tweet): Status = {

assert(tweet.coreData.nonEmpty, "tweet core data is missing")

val coreData = tweet.coreData.get

val toGeo: Option[Geo] =

coreData.coordinates match {

case Some(coords) =>

Some(

Geo(

latitude = coords.latitude,

longitude = coords.longitude,

geoPrecision = coords.geoPrecision,

entityId = if (coords.display) 2 else 0,

name = coreData.placeId,

place = tweet.place,

placeId = coreData.placeId,

coordinates = Some(coords)

)

)

case \_ =>

coreData.placeId match {

case None => None

case Some(\_) =>

Some(Geo(name = coreData.placeId, place = tweet.place, placeId = coreData.placeId))

}

}

Status(

id = tweet.id,

userId = coreData.userId,

text = coreData.text,

createdVia = coreData.createdVia,

createdAt = coreData.createdAtSecs,

urls = tweet.urls.getOrElse(Seq.empty),

mentions = tweet.mentions.getOrElse(Seq.empty),

hashtags = tweet.hashtags.getOrElse(Seq.empty),

cashtags = tweet.cashtags.getOrElse(Seq.empty),

media = tweet.media.getOrElse(Seq.empty),

reply = tweet.coreData.flatMap(\_.reply),

directedAtUser = tweet.coreData.flatMap(\_.directedAtUser),

share = tweet.coreData.flatMap(\_.share),

quotedTweet = tweet.quotedTweet,

geo = toGeo,

hasTakedown = coreData.hasTakedown,

nsfwUser = coreData.nsfwUser,

nsfwAdmin = coreData.nsfwAdmin,

counts = tweet.counts,

deviceSource = tweet.deviceSource,

narrowcast = coreData.narrowcast,

takedownCountryCodes = tweet.takedownCountryCodes,

perspective = tweet.perspective,

cards = tweet.cards,

card2 = tweet.card2,

nullcast = coreData.nullcast,

conversationId = coreData.conversationId,

language = tweet.language,

trackingId = coreData.trackingId,

spamLabels = tweet.spamLabels,

hasMedia = coreData.hasMedia,

contributor = tweet.contributor,

mediaTags = tweet.mediaTags

)

}

def toTweet(status: Status): Tweet = {

val coreData =

TweetCoreData(

userId = status.userId,

text = status.text,

createdVia = status.createdVia,

createdAtSecs = status.createdAt,

reply = status.reply,

directedAtUser = status.directedAtUser,

share = status.share,

hasTakedown = status.hasTakedown,

nsfwUser = status.nsfwUser,

nsfwAdmin = status.nsfwAdmin,

nullcast = status.nullcast,

narrowcast = status.narrowcast,

trackingId = status.trackingId,

conversationId = status.conversationId,

hasMedia = status.hasMedia,

coordinates = toCoords(status),

placeId = status.geo.flatMap(\_.placeId)

)

Tweet(

id = status.id,

coreData = Some(coreData),

urls = Some(status.urls),

mentions = Some(status.mentions),

hashtags = Some(status.hashtags),

cashtags = Some(status.cashtags),

media = Some(status.media),

place = status.geo.flatMap(\_.place),

quotedTweet = status.quotedTweet,

takedownCountryCodes = status.takedownCountryCodes,

counts = status.counts,

deviceSource = status.deviceSource,

perspective = status.perspective,

cards = status.cards,

card2 = status.card2,

language = status.language,

spamLabels = status.spamLabels,

contributor = status.contributor,

mediaTags = status.mediaTags

)

}

private def toCoords(status: Status): Option[GeoCoordinates] =

status.geo.map { geo =>

if (geo.coordinates.nonEmpty) geo.coordinates.get

// Status from monorail have the coordinates as the top level fields in Geo,

// while the nested struct is empty. So we need to copy from the flat fields.

else

GeoCoordinates(

latitude = geo.latitude,

longitude = geo.longitude,

geoPrecision = geo.geoPrecision,

display = geo.entityId == 2

)

}

}