package com.twitter.tweetypie.hydrator

import com.twitter.mediaservices.commons.tweetmedia.thriftscala.\_

import com.twitter.mediaservices.commons.thriftscala.\_

import com.twitter.tweetypie.core.ValueState

import com.twitter.tweetypie.thriftscala.\_

object MediaKeyHydrator {

type Ctx = MediaEntityHydrator.Uncacheable.Ctx

type Type = MediaEntityHydrator.Uncacheable.Type

def apply(): Type =

ValueHydrator

.map[MediaEntity, Ctx] { (curr, ctx) =>

val mediaKey = infer(ctx.mediaKeys, curr)

ValueState.modified(curr.copy(mediaKey = Some(mediaKey)))

}

.onlyIf((curr, ctx) => curr.mediaKey.isEmpty)

def infer(mediaKeys: Option[Seq[MediaKey]], mediaEntity: MediaEntity): MediaKey = {

def inferByMediaId =

mediaKeys

.flatMap(\_.find(\_.mediaId == mediaEntity.mediaId))

def contentType =

mediaEntity.sizes.find(\_.sizeType == MediaSizeType.Orig).map(\_.deprecatedContentType)

def inferByContentType =

contentType.map { tpe =>

val category =

tpe match {

case MediaContentType.VideoMp4 => MediaCategory.TweetGif

case MediaContentType.VideoGeneric => MediaCategory.TweetVideo

case \_ => MediaCategory.TweetImage

}

MediaKey(category, mediaEntity.mediaId)

}

def fail =

throw new IllegalStateException(

s"""

|Can't infer media key.

| mediaKeys:'$mediaKeys'

| mediaEntity:'$mediaEntity'

""".stripMargin

)

mediaEntity.mediaKey

.orElse(inferByMediaId)

.orElse(inferByContentType)

.getOrElse(fail)

}

}