package com.twitter.tweetypie

package hydrator

import com.twitter.stitch.Stitch

import com.twitter.tweetypie.core.\_

import com.twitter.tweetypie.media.MediaKeyUtil

import com.twitter.tweetypie.media.MediaMetadataRequest

import com.twitter.tweetypie.repository.\_

import com.twitter.tweetypie.thriftscala.\_

import java.nio.ByteBuffer

object MediaInfoHydrator {

type Ctx = MediaEntityHydrator.Uncacheable.Ctx

type Type = MediaEntityHydrator.Uncacheable.Type

private[this] val log = Logger(getClass)

def apply(repo: MediaMetadataRepository.Type, stats: StatsReceiver): Type = {

val attributableUserCounter = stats.counter("attributable\_user")

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

val request =

toMediaMetadataRequest(

mediaEntity = curr,

tweetId = ctx.tweetId,

extensionsArgs = ctx.opts.extensionsArgs

)

request match {

case None => Stitch.value(ValueState.unmodified(curr))

case Some(req) =>

repo(req).liftToTry.map {

case Return(metadata) =>

if (metadata.attributableUserId.nonEmpty) attributableUserCounter.incr()

ValueState.delta(

curr,

metadata.updateEntity(

mediaEntity = curr,

tweetUserId = ctx.userId,

includeAdditionalMetadata = ctx.includeAdditionalMetadata

)

)

case Throw(ex) if !PartialEntityCleaner.isPartialMedia(curr) =>

log.info("Ignored media info repo failure, media entity already hydrated", ex)

ValueState.unmodified(curr)

case Throw(ex) =>

log.error("Media info hydration failed", ex)

ValueState.partial(curr, MediaEntityHydrator.hydratedField)

}

}

}

}

def toMediaMetadataRequest(

mediaEntity: MediaEntity,

tweetId: TweetId,

extensionsArgs: Option[ByteBuffer]

): Option[MediaMetadataRequest] =

mediaEntity.isProtected.map { isProtected =>

val mediaKey = MediaKeyUtil.get(mediaEntity)

MediaMetadataRequest(

tweetId = tweetId,

mediaKey = mediaKey,

isProtected = isProtected,

extensionsArgs = extensionsArgs

)

}

}