package com.twitter.tweetypie

package media

import com.twitter.mediaservices.commons.mediainformation.{thriftscala => mic}

import com.twitter.mediaservices.commons.thriftscala.MediaKey

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

import com.twitter.tweetypie.thriftscala.\_

import java.nio.ByteBuffer

/\*\*

\* MediaMetadata encapsulates the metadata about tweet media that we receive from

\* the various media services backends on tweet create or on tweet read. This data,

\* combined with data stored on the tweet, is sufficient to hydrate tweet media entities.

\*/

case class MediaMetadata(

mediaKey: MediaKey,

assetUrlHttps: String,

sizes: Set[MediaSize],

mediaInfo: MediaInfo,

productMetadata: Option[mic.UserDefinedProductMetadata] = None,

extensionsReply: Option[ByteBuffer] = None,

additionalMetadata: Option[mic.AdditionalMetadata] = None) {

def assetUrlHttp: String = MediaUrl.httpsToHttp(assetUrlHttps)

def attributableUserId: Option[UserId] =

additionalMetadata.flatMap(\_.ownershipInfo).flatMap(\_.attributableUserId)

def updateEntity(

mediaEntity: MediaEntity,

tweetUserId: UserId,

includeAdditionalMetadata: Boolean

): MediaEntity = {

// Abort if we accidentally try to replace the media. This

// indicates a logic error that caused mismatched media info.

// This could be internal or external to TweetyPie.

require(

mediaEntity.mediaId == mediaKey.mediaId,

"Tried to update media with mediaId=%s with mediaInfo.mediaId=%s"

.format(mediaEntity.mediaId, mediaKey.mediaId)

)

mediaEntity.copy(

mediaUrl = assetUrlHttp,

mediaUrlHttps = assetUrlHttps,

sizes = sizes,

mediaInfo = Some(mediaInfo),

extensionsReply = extensionsReply,

// the following two fields are deprecated and will be removed soon

nsfw = false,

mediaPath = MediaUrl.mediaPathFromUrl(assetUrlHttps),

metadata = productMetadata,

additionalMetadata = additionalMetadata.filter(\_ => includeAdditionalMetadata),

// MIS allows media to be shared among authorized users so add in sourceUserId if it doesn't

// match the current tweet's userId.

sourceUserId = attributableUserId.filter(\_ != tweetUserId)

)

}

}