package com.twitter.tweetypie

package media

import com.twitter.mediaservices.commons.thriftscala.MediaCategory

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

import com.twitter.tco\_util.TcoSlug

import com.twitter.tweetypie.thriftscala.\_

import com.twitter.tweetypie.util.TweetLenses

/\*\*

\* A smörgåsbord of media-related helper methods.

\*/

object Media {

val AnimatedGifContentType = "video/mp4 codecs=avc1.42E0"

case class MediaTco(expandedUrl: String, url: String, displayUrl: String)

val ImageContentTypes: Set[MediaContentType] =

Set[MediaContentType](

MediaContentType.ImageJpeg,

MediaContentType.ImagePng,

MediaContentType.ImageGif

)

val AnimatedGifContentTypes: Set[MediaContentType] =

Set[MediaContentType](

MediaContentType.VideoMp4

)

val VideoContentTypes: Set[MediaContentType] =

Set[MediaContentType](

MediaContentType.VideoGeneric

)

val InUseContentTypes: Set[MediaContentType] =

Set[MediaContentType](

MediaContentType.ImageGif,

MediaContentType.ImageJpeg,

MediaContentType.ImagePng,

MediaContentType.VideoMp4,

MediaContentType.VideoGeneric

)

def isImage(contentType: MediaContentType): Boolean =

ImageContentTypes.contains(contentType)

def contentTypeToString(contentType: MediaContentType): String =

contentType match {

case MediaContentType.ImageGif => "image/gif"

case MediaContentType.ImageJpeg => "image/jpeg"

case MediaContentType.ImagePng => "image/png"

case MediaContentType.VideoMp4 => "video/mp4"

case MediaContentType.VideoGeneric => "video"

case \_ => throw new IllegalArgumentException(s"UnknownMediaContentType: $contentType")

}

def stringToContentType(str: String): MediaContentType =

str match {

case "image/gif" => MediaContentType.ImageGif

case "image/jpeg" => MediaContentType.ImageJpeg

case "image/png" => MediaContentType.ImagePng

case "video/mp4" => MediaContentType.VideoMp4

case "video" => MediaContentType.VideoGeneric

case \_ => throw new IllegalArgumentException(s"Unknown Content Type String: $str")

}

def extensionForContentType(cType: MediaContentType): String =

cType match {

case MediaContentType.ImageJpeg => "jpg"

case MediaContentType.ImagePng => "png"

case MediaContentType.ImageGif => "gif"

case MediaContentType.VideoMp4 => "mp4"

case MediaContentType.VideoGeneric => ""

case \_ => "unknown"

}

/\*\*

\* Extract a URL entity from a media entity.

\*/

def extractUrlEntity(mediaEntity: MediaEntity): UrlEntity =

UrlEntity(

fromIndex = mediaEntity.fromIndex,

toIndex = mediaEntity.toIndex,

url = mediaEntity.url,

expanded = Some(mediaEntity.expandedUrl),

display = Some(mediaEntity.displayUrl)

)

/\*\*

\* Copy the fields from the URL entity into the media entity.

\*/

def copyFromUrlEntity(mediaEntity: MediaEntity, urlEntity: UrlEntity): MediaEntity = {

val expandedUrl =

urlEntity.expanded.orElse(Option(mediaEntity.expandedUrl)).getOrElse(urlEntity.url)

val displayUrl =

urlEntity.url match {

case TcoSlug(slug) => MediaUrl.Display.fromTcoSlug(slug)

case \_ => urlEntity.expanded.getOrElse(urlEntity.url)

}

mediaEntity.copy(

fromIndex = urlEntity.fromIndex,

toIndex = urlEntity.toIndex,

url = urlEntity.url,

expandedUrl = expandedUrl,

displayUrl = displayUrl

)

}

def getAspectRatio(size: MediaSize): AspectRatio =

getAspectRatio(size.width, size.height)

def getAspectRatio(width: Int, height: Int): AspectRatio = {

if (width == 0 || height == 0) {

throw new IllegalArgumentException(s"Dimensions must be non zero: ($width, $height)")

}

def calculateGcd(a: Int, b: Int): Int =

if (b == 0) a else calculateGcd(b, a % b)

val gcd = calculateGcd(math.max(width, height), math.min(width, height))

AspectRatio((width / gcd).toShort, (height / gcd).toShort)

}

/\*\*

\* Return just the media that belongs to this tweet

\*/

def ownMedia(tweet: Tweet): Seq[MediaEntity] =

TweetLenses.media.get(tweet).filter(isOwnMedia(tweet.id, \_))

/\*\*

\* Does the given media entity, which is was found on the tweet with the specified

\* tweetId, belong to that tweet?

\*/

def isOwnMedia(tweetId: TweetId, entity: MediaEntity): Boolean =

entity.sourceStatusId.forall(\_ == tweetId)

/\*\*

\* Mixed Media is any case where there is more than one media item & any of them is not an image.

\*/

def isMixedMedia(mediaEntities: Seq[MediaEntity]): Boolean =

mediaEntities.length > 1 && (mediaEntities.flatMap(\_.mediaInfo).exists {

case \_: MediaInfo.ImageInfo => false

case \_ => true

} ||

mediaEntities.flatMap(\_.mediaKey).map(\_.mediaCategory).exists(\_ != MediaCategory.TweetImage))

}