package com.twitter.tweetypie

package media

import com.twitter.logging.Logger

import com.twitter.tweetypie.thriftscala.MediaEntity

import com.twitter.tweetypie.thriftscala.UrlEntity

/\*\*

\* Creating and parsing tweet media entity URLs.

\*

\* There are four kinds of URL in a media entity:

\*

\* - Display URLs: pic.twitter.com aliases for the short URL, for

\* embedding in the tweet text.

\*

\* - Short URLs: regular t.co URLs that expand to the permalink URL.

\*

\* - Permalink URLs: link to a page that displays the media after

\* doing authorization

\*

\* - Asset URLs: links to the actual media asset.

\*

\*/

object MediaUrl {

private[this] val log = Logger(getClass)

/\*\*

\* The URL that should be filled in to the displayUrl field of the

\* media entity. This URL behaves exactly the same as a t.co link

\* (only the domain is different.)

\*/

object Display {

val Root = "pic.twitter.com/"

def fromTcoSlug(tcoSlug: String): String = Root + tcoSlug

}

/\*\*

\* The link target for the link in the tweet text (the expanded URL

\* for the media, copied from the URL entity.) For native photos,

\* this is the tweet permalink page.

\*

\* For users without a screen name ("handleless" or NoScreenName users)

\* a permalink to /i/status/:tweet\_id is used.

\*/

object Permalink {

val Root = "https://twitter.com/"

val Internal = "i"

val PhotoSuffix = "/photo/1"

val VideoSuffix = "/video/1"

def apply(screenName: String, tweetId: TweetId, isVideo: Boolean): String =

Root +

(if (screenName.isEmpty) Internal else screenName) +

"/status/" +

tweetId +

(if (isVideo) VideoSuffix else PhotoSuffix)

private[this] val PermalinkRegex =

"""https?://twitter.com/(?:#!/)?\w+/status/(\d+)/(?:photo|video)/\d+""".r

private[this] def getTweetId(permalink: String): Option[TweetId] =

permalink match {

case PermalinkRegex(tweetIdStr) =>

try {

Some(tweetIdStr.toLong)

} catch {

// Digits too big to fit in a Long

case \_: NumberFormatException => None

}

case \_ => None

}

def getTweetId(urlEntity: UrlEntity): Option[TweetId] =

urlEntity.expanded.flatMap(getTweetId)

def hasTweetId(permalink: String, tweetId: TweetId): Boolean =

getTweetId(permalink).contains(tweetId)

def hasTweetId(mediaEntity: MediaEntity, tweetId: TweetId): Boolean =

hasTweetId(mediaEntity.expandedUrl, tweetId)

def hasTweetId(urlEntity: UrlEntity, tweetId: TweetId): Boolean =

getTweetId(urlEntity).contains(tweetId)

}

/\*\*

\* Converts a url that starts with "https://" to one that starts with "http://".

\*/

def httpsToHttp(url: String): String =

url.replace("https://", "http://")

/\*\*

\* Gets the last path element from an asset url. This exists temporarily to support

\* the now deprecated mediaPath element in MediaEntity.

\*/

def mediaPathFromUrl(url: String): String =

url.lastIndexOf('/') match {

case -1 =>

log.error("Invalid media path. Could not find last element: " + url)

// Better to return a broken preview URL to the client

// than to fail the whole request.

""

case idx =>

url.substring(idx + 1)

}

}