package com.twitter.tweetypie

package handler

import com.twitter.mediaservices.commons.mediainformation.thriftscala.UserDefinedProductMetadata

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

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

import com.twitter.servo.util.FutureArrow

import com.twitter.tco\_util.TcoSlug

import com.twitter.tweetypie.core.\_

import com.twitter.tweetypie.media.\_

import com.twitter.tweetypie.serverutil.ExceptionCounter

import com.twitter.tweetypie.thriftscala.\_

import com.twitter.tweetypie.tweettext.Offset

object CreateMediaTco {

import UpstreamFailure.\_

case class Request(

tweetId: TweetId,

userId: UserId,

userScreenName: String,

isProtected: Boolean,

createdAt: Time,

isVideo: Boolean,

dark: Boolean)

type Type = FutureArrow[Request, Media.MediaTco]

def apply(urlShortener: UrlShortener.Type): Type =

FutureArrow[Request, Media.MediaTco] { req =>

val expandedUrl = MediaUrl.Permalink(req.userScreenName, req.tweetId, req.isVideo)

val shortenCtx =

UrlShortener.Context(

userId = req.userId,

userProtected = req.isProtected,

tweetId = req.tweetId,

createdAt = req.createdAt,

dark = req.dark

)

urlShortener((expandedUrl, shortenCtx))

.flatMap { metadata =>

metadata.shortUrl match {

case TcoSlug(slug) =>

Future.value(

Media.MediaTco(

expandedUrl,

metadata.shortUrl,

MediaUrl.Display.fromTcoSlug(slug)

)

)

case \_ =>

// should never get here, since shortened urls from talon

// always start with "http://t.co/", just in case...

Future.exception(MediaShortenUrlMalformedFailure)

}

}

.rescue {

case UrlShortener.InvalidUrlError =>

// should never get here, since media expandedUrl should always be a valid

// input to talon.

Future.exception(MediaExpandedUrlNotValidFailure)

}

}

}

object MediaBuilder {

private val log = Logger(getClass)

case class Request(

mediaUploadIds: Seq[MediaId],

text: String,

tweetId: TweetId,

userId: UserId,

userScreenName: String,

isProtected: Boolean,

createdAt: Time,

dark: Boolean = false,

productMetadata: Option[Map[MediaId, UserDefinedProductMetadata]] = None)

case class Result(updatedText: String, mediaEntities: Seq[MediaEntity], mediaKeys: Seq[MediaKey])

type Type = FutureArrow[Request, Result]

def apply(

processMedia: MediaClient.ProcessMedia,

createMediaTco: CreateMediaTco.Type,

stats: StatsReceiver

): Type =

FutureArrow[Request, Result] {

case Request(

mediaUploadIds,

text,

tweetId,

userId,

screenName,

isProtected,

createdAt,

dark,

productMetadata

) =>

for {

mediaKeys <- processMedia(

ProcessMediaRequest(

mediaUploadIds,

userId,

tweetId,

isProtected,

productMetadata

)

)

mediaTco <- createMediaTco(

CreateMediaTco.Request(

tweetId,

userId,

screenName,

isProtected,

createdAt,

mediaKeys.exists(MediaKeyClassifier.isVideo(\_)),

dark

)

)

} yield produceResult(text, mediaTco, isProtected, mediaKeys)

}.countExceptions(

ExceptionCounter(stats)

)

.onFailure[Request] { (req, ex) => log.info(req.toString, ex) }

.translateExceptions {

case e: MediaExceptions.MediaClientException =>

TweetCreateFailure.State(TweetCreateState.InvalidMedia, Some(e.getMessage))

}

def produceResult(

text: String,

mediaTco: Media.MediaTco,

userIsProtected: Boolean,

mediaKeys: Seq[MediaKey]

): Result = {

val newText =

if (text == "") mediaTco.url

else text + " " + mediaTco.url

val to = Offset.CodePoint.length(newText)

val from = to - Offset.CodePoint.length(mediaTco.url)

val mediaEntities =

mediaKeys.map { mediaKey =>

MediaEntity(

mediaKey = Some(mediaKey),

fromIndex = from.toShort,

toIndex = to.toShort,

url = mediaTco.url,

displayUrl = mediaTco.displayUrl,

expandedUrl = mediaTco.expandedUrl,

mediaId = mediaKey.mediaId,

mediaPath = "", // to be hydrated

mediaUrl = null, // to be hydrated

mediaUrlHttps = null, // to be hydrated

nsfw = false, // deprecated

sizes = Set(

MediaSize(

sizeType = MediaSizeType.Orig,

resizeMethod = MediaResizeMethod.Fit,

deprecatedContentType = MediaKeyUtil.contentType(mediaKey),

width = -1, // to be hydrated

height = -1 // to be hydrated

)

)

)

}

Result(newText, mediaEntities, mediaKeys)

}

}