package com.twitter.tweetypie

package handler

import com.twitter.expandodo.thriftscala.AttachmentEligibilityResponses

import com.twitter.expandodo.{thriftscala => expandodo}

import com.twitter.tweetypie.backends.Expandodo

import com.twitter.twittertext.Extractor

import scala.util.control.NoStackTrace

import scala.util.control.NonFatal

import java.net.URI

object CardReferenceValidationFailedException extends Exception with NoStackTrace

object CardReferenceValidationHandler {

type Type = FutureArrow[(UserId, CardUri), CardUri]

def apply(checkEligibility: Expandodo.CheckAttachmentEligibility): Type = {

def validateAttachmentForUser(userId: UserId, cardUri: CardUri): Future[CardUri] = {

val request = Seq(expandodo.AttachmentEligibilityRequest(cardUri, userId))

checkEligibility(request)

.flatMap(validatedCardUri)

.rescue {

case NonFatal(\_) => Future.exception(CardReferenceValidationFailedException)

}

}

FutureArrow {

case (userId, cardUri) =>

if (shouldSkipValidation(cardUri)) {

Future.value(cardUri)

} else {

validateAttachmentForUser(userId, cardUri)

}

}

}

private[this] def validatedCardUri(responses: AttachmentEligibilityResponses) = {

responses.results.headOption match {

case Some(

expandodo.AttachmentEligibilityResult

.Success(expandodo.ValidCardUri(validatedCardUri))

) =>

Future.value(validatedCardUri)

case \_ =>

Future.exception(CardReferenceValidationFailedException)

}

}

// We're not changing state between calls, so it's safe to share among threads

private[this] val extractor = {

val extractor = new Extractor

extractor.setExtractURLWithoutProtocol(false)

extractor

}

// Card References with these URIs don't need validation since cards referenced by URIs in these

// schemes are public and hence not subject to restrictions.

private[handler] val isWhitelistedSchema = Set("http", "https", "tombstone")

// NOTE: http://www.ietf.org/rfc/rfc2396.txt

private[this] def hasWhitelistedScheme(cardUri: CardUri) =

Try(new URI(cardUri)).toOption

.map(\_.getScheme)

.exists(isWhitelistedSchema)

// Even though URI spec is technically is a superset of http:// and https:// URLs, we have to

// resort to using a Regex based parser here as a fallback because many URLs found in the wild

// have unescaped components that would fail java.net.URI parsing, yet are still considered acceptable.

private[this] def isTwitterUrlEntity(cardUri: CardUri) =

extractor.extractURLs(cardUri).size == 1

private[this] def shouldSkipValidation(cardUri: CardUri) =

hasWhitelistedScheme(cardUri) || isTwitterUrlEntity(cardUri)

}