package com.twitter.tweetypie

package handler

import com.twitter.servo.util.FutureArrow

import com.twitter.tweetypie.backends.LimiterService

import com.twitter.tweetypie.core.TweetCreateFailure

import com.twitter.tweetypie.thriftscala.TweetCreateState.RateLimitExceeded

object RateLimitChecker {

type Dark = Boolean

type GetRemaining = FutureArrow[(UserId, Dark), Int]

type Validate = FutureArrow[(UserId, Dark), Unit]

def getMaxMediaTags(minRemaining: LimiterService.MinRemaining, maxMediaTags: Int): GetRemaining =

FutureArrow {

case (userId, dark) =>

if (dark) Future.value(maxMediaTags)

else {

val contributorUserId = getContributor(userId).map(\_.userId)

minRemaining(userId, contributorUserId)

.map(\_.min(maxMediaTags))

.handle { case \_ => maxMediaTags }

}

}

def validate(

hasRemaining: LimiterService.HasRemaining,

featureStats: StatsReceiver,

rateLimitEnabled: () => Boolean

): Validate = {

val exceededCounter = featureStats.counter("exceeded")

val checkedCounter = featureStats.counter("checked")

FutureArrow {

case (userId, dark) =>

if (dark || !rateLimitEnabled()) {

Future.Unit

} else {

checkedCounter.incr()

val contributorUserId = getContributor(userId).map(\_.userId)

hasRemaining(userId, contributorUserId).map {

case false =>

exceededCounter.incr()

throw TweetCreateFailure.State(RateLimitExceeded)

case \_ => ()

}

}

}

}

}