package com.twitter.tweetypie.handler

import com.twitter.featureswitches.v2.FeatureSwitchResults

import com.twitter.servo.util.Gate

import com.twitter.tweetypie.Future

import com.twitter.tweetypie.core.TweetCreateFailure

import com.twitter.tweetypie.thriftscala.Communities

import com.twitter.tweetypie.thriftscala.TweetCreateState.CommunityProtectedUserCannotTweet

import com.twitter.tweetypie.util.CommunityUtil

object CommunitiesValidator {

case class Request(

matchedResults: Option[FeatureSwitchResults],

isProtected: Boolean,

community: Option[Communities])

type Type = Request => Future[Unit]

val CommunityProtectedCanCreateTweet = "communities\_protected\_community\_tweet\_creation\_enabled"

val communityProtectedCanCreateTweetGate: Gate[Request] = Gate { request: Request =>

request.matchedResults

.flatMap(\_.getBoolean(CommunityProtectedCanCreateTweet, shouldLogImpression = true))

.contains(false)

}

def apply(): Type =

(request: Request) => {

// Order is important: the feature-switch gate is checked only when the

// request is both protected & community so that the FS experiment measurements

// are based only on data from requests that are subject to rejection by this validator.

if (request.isProtected &&

CommunityUtil.hasCommunity(request.community) &&

communityProtectedCanCreateTweetGate(request)) {

Future.exception(TweetCreateFailure.State(CommunityProtectedUserCannotTweet))

} else {

Future.Unit

}

}

}