package com.twitter.tweetypie

package handler

import com.twitter.finagle.tracing.Trace

import com.twitter.service.gen.scarecrow.thriftscala.Retweet

import com.twitter.service.gen.scarecrow.thriftscala.TieredAction

import com.twitter.service.gen.scarecrow.thriftscala.TieredActionResult

import com.twitter.spam.features.thriftscala.SafetyMetaData

import com.twitter.stitch.Stitch

import com.twitter.tweetypie.core.TweetCreateFailure

import com.twitter.tweetypie.repository.RetweetSpamCheckRepository

import com.twitter.tweetypie.thriftscala.TweetCreateState

case class RetweetSpamRequest(

retweetId: TweetId,

sourceUserId: UserId,

sourceTweetId: TweetId,

sourceTweetText: String,

sourceUserName: Option[String],

safetyMetaData: Option[SafetyMetaData])

/\*\*

\* Use the Scarecrow service as the spam checker for retweets.

\*/

object ScarecrowRetweetSpamChecker {

val log: Logger = Logger(getClass)

def requestToScarecrowRetweet(req: RetweetSpamRequest): Retweet =

Retweet(

id = req.retweetId,

sourceUserId = req.sourceUserId,

text = req.sourceTweetText,

sourceTweetId = req.sourceTweetId,

safetyMetaData = req.safetyMetaData

)

def apply(

stats: StatsReceiver,

repo: RetweetSpamCheckRepository.Type

): Spam.Checker[RetweetSpamRequest] = {

def handler(request: RetweetSpamRequest): Spam.Checker[TieredAction] =

Spam.handleScarecrowResult(stats) {

case (TieredActionResult.NotSpam, \_, \_) => Spam.AllowFuture

case (TieredActionResult.SilentFail, \_, \_) => Spam.SilentFailFuture

case (TieredActionResult.UrlSpam, \_, denyMessage) =>

Future.exception(TweetCreateFailure.State(TweetCreateState.UrlSpam, denyMessage))

case (TieredActionResult.Deny, \_, denyMessage) =>

Future.exception(TweetCreateFailure.State(TweetCreateState.Spam, denyMessage))

case (TieredActionResult.DenyByIpiPolicy, \_, denyMessage) =>

Future.exception(Spam.DisabledByIpiFailure(request.sourceUserName, denyMessage))

case (TieredActionResult.RateLimit, \_, denyMessage) =>

Future.exception(

TweetCreateFailure.State(TweetCreateState.SafetyRateLimitExceeded, denyMessage))

case (TieredActionResult.Bounce, Some(b), \_) =>

Future.exception(TweetCreateFailure.Bounced(b))

}

req => {

Trace.record("com.twitter.tweetypie.ScarecrowRetweetSpamChecker.retweetId=" + req.retweetId)

Stitch.run(repo(requestToScarecrowRetweet(req))).flatMap(handler(req))

}

}

}