package com.twitter.tweetypie

package repository

import com.twitter.context.thriftscala.Viewer

import com.twitter.featureswitches.Recipient

import com.twitter.featureswitches.TOOClient

import com.twitter.featureswitches.UserAgent

import com.twitter.tweetypie.StatsReceiver

import com.twitter.tweetypie.User

import com.twitter.tweetypie.UserId

import com.twitter.tweetypie.client\_id.ClientIdHelper

import com.twitter.tweetypie.repository.UserViewerRecipient.UserIdMismatchException

/\*\*

\* Provides a Recipient backed by a Gizmoduck User and TwitterContext Viewer for

\* use in FeatureSwitch validation.

\*/

object UserViewerRecipient {

object UserIdMismatchException extends Exception

def apply(user: User, viewer: Viewer, stats: StatsReceiver): Option[Recipient] = {

// This is a workaround for thrift API clients that allow users to Tweet on behalf

// of other Twitter users. This is similar to go/contributors, however some platforms

// have enabled workflows that don't use the go/contributors auth platform, and

// therefore the TwitterContext Viewer isn't set up correctly for contributor requests.

if (viewer.userId.contains(user.id)) {

Some(new UserViewerRecipient(user, viewer))

} else {

val mismatchScope = stats.scope(s"user\_viewer\_mismatch")

ClientIdHelper.default.effectiveClientIdRoot.foreach { clientId =>

mismatchScope.scope("client").counter(clientId).incr()

}

mismatchScope.counter("total").incr()

None

}

}

}

class UserViewerRecipient(

user: User,

viewer: Viewer)

extends Recipient {

if (!viewer.userId.contains(user.id)) {

throw UserIdMismatchException

}

override def userId: Option[UserId] = viewer.userId

override def userRoles: Option[Set[String]] = user.roles.map(\_.roles.toSet)

override def deviceId: Option[String] = viewer.deviceId

override def guestId: Option[Long] = viewer.guestId

override def languageCode: Option[String] = viewer.requestLanguageCode

override def signupCountryCode: Option[String] = user.safety.flatMap(\_.signupCountryCode)

override def countryCode: Option[String] = viewer.requestCountryCode

override def userAgent: Option[UserAgent] = viewer.userAgent.flatMap(UserAgent(\_))

override def isManifest: Boolean = false

override def isVerified: Option[Boolean] = user.safety.map(\_.verified)

override def clientApplicationId: Option[Long] = viewer.clientApplicationId

@Deprecated

override def isTwoffice: Option[Boolean] = None

@Deprecated

override def tooClient: Option[TOOClient] = None

@Deprecated

override def highWaterMark: Option[Long] = None

}