package com.twitter.tweetypie

import com.twitter.context.thriftscala.Viewer

import com.twitter.tweetypie.thriftscala.\_

import scala.util.matching.Regex

import com.twitter.context.TwitterContext

import com.twitter.finagle.stats.Stat

import com.twitter.snowflake.id.SnowflakeId

package object handler {

type PlaceLanguage = String

type TweetIdGenerator = () => Future[TweetId]

type NarrowcastValidator = FutureArrow[Narrowcast, Narrowcast]

type ReverseGeocoder = FutureArrow[(GeoCoordinates, PlaceLanguage), Option[Place]]

type CardUri = String

// A narrowcast location can be a PlaceId or a US metro code.

type NarrowcastLocation = String

val PlaceIdRegex: Regex = """(?i)\A[0-9a-fA-F]{16}\Z""".r

// Bring Tweetypie permitted TwitterContext into scope

val TwitterContext: TwitterContext =

com.twitter.context.TwitterContext(com.twitter.tweetypie.TwitterContextPermit)

def getContributor(userId: UserId): Option[Contributor] = {

val viewer = TwitterContext().getOrElse(Viewer())

viewer.authenticatedUserId.filterNot(\_ == userId).map(id => Contributor(id))

}

def trackLossyReadsAfterWrite(stat: Stat, windowLength: Duration)(tweetId: TweetId): Unit = {

// If the requested Tweet is NotFound, and the tweet age is less than the defined {{windowLength}} duration,

// then we capture the percentiles of when this request was attempted.

// This is being tracked to understand how lossy the reads are directly after tweet creation.

for {

timestamp <- SnowflakeId.timeFromIdOpt(tweetId)

age = Time.now.since(timestamp)

if age.inMillis <= windowLength.inMillis

} yield stat.add(age.inMillis)

}

}