package com.twitter.tweetypie

package handler

import com.twitter.stitch.Stitch

import com.twitter.tweetypie.repository.\_

import com.twitter.tweetypie.store.ScrubGeo

import com.twitter.tweetypie.store.ScrubGeoUpdateUserTimestamp

import com.twitter.tweetypie.thriftscala.DeleteLocationData

import com.twitter.tweetypie.thriftscala.GeoScrub

/\*\*

\* Create the appropriate ScrubGeo.Event for a GeoScrub request.

\*/

object ScrubGeoEventBuilder {

val userQueryOptions: UserQueryOptions =

UserQueryOptions(

Set(UserField.Safety, UserField.Roles),

UserVisibility.All

)

private def userLoader(

stats: StatsReceiver,

userRepo: UserRepository.Optional

): UserId => Future[Option[User]] = {

val userNotFoundCounter = stats.counter("user\_not\_found")

userId =>

Stitch.run(

userRepo(UserKey(userId), userQueryOptions)

.onSuccess(userOpt => if (userOpt.isEmpty) userNotFoundCounter.incr())

)

}

object UpdateUserTimestamp {

type Type = DeleteLocationData => Future[ScrubGeoUpdateUserTimestamp.Event]

def apply(

stats: StatsReceiver,

userRepo: UserRepository.Optional,

): Type = {

val timestampDiffStat = stats.stat("now\_delta\_ms")

val loadUser = userLoader(stats, userRepo)

request: DeleteLocationData =>

loadUser(request.userId).map { userOpt =>

// delta between users requesting deletion and the time we publish to TweetEvents

timestampDiffStat.add((Time.now.inMillis - request.timestampMs).toFloat)

ScrubGeoUpdateUserTimestamp.Event(

userId = request.userId,

timestamp = Time.fromMilliseconds(request.timestampMs),

optUser = userOpt

)

}

}

}

object ScrubTweets {

type Type = GeoScrub => Future[ScrubGeo.Event]

def apply(stats: StatsReceiver, userRepo: UserRepository.Optional): Type = {

val loadUser = userLoader(stats, userRepo)

geoScrub =>

loadUser(geoScrub.userId).map { userOpt =>

ScrubGeo.Event(

tweetIdSet = geoScrub.statusIds.toSet,

userId = geoScrub.userId,

enqueueMax = geoScrub.hosebirdEnqueue,

optUser = userOpt,

timestamp = Time.now

)

}

}

}

}