package com.twitter.tweetypie

package store

import com.twitter.gizmoduck.thriftscala.LookupContext

import com.twitter.gizmoduck.thriftscala.ModifiedAccount

import com.twitter.gizmoduck.thriftscala.ModifiedUser

import com.twitter.tweetypie.backends.Gizmoduck

import com.twitter.tweetypie.thriftscala.\_

trait GizmoduckUserGeotagUpdateStore

extends TweetStoreBase[GizmoduckUserGeotagUpdateStore]

with AsyncInsertTweet.Store

with ScrubGeoUpdateUserTimestamp.Store {

def wrap(w: TweetStore.Wrap): GizmoduckUserGeotagUpdateStore =

new TweetStoreWrapper(w, this)

with GizmoduckUserGeotagUpdateStore

with AsyncInsertTweet.StoreWrapper

with ScrubGeoUpdateUserTimestamp.StoreWrapper

}

/\*\*

\* A TweetStore implementation that updates a Gizmoduck user's user\_has\_geotagged\_status flag.

\* If a tweet is geotagged and the user's flag is not set, call out to Gizmoduck to update it.

\*/

object GizmoduckUserGeotagUpdateStore {

val Action: AsyncWriteAction.UserGeotagUpdate.type = AsyncWriteAction.UserGeotagUpdate

def apply(

modifyAndGet: Gizmoduck.ModifyAndGet,

stats: StatsReceiver

): GizmoduckUserGeotagUpdateStore = {

// Counts the number of times that the scrubGeo actually cleared the

// hasGeotaggedStatuses bit for a user.

val clearedCounter = stats.counter("has\_geotag\_cleared")

// Counts the number of times that asyncInsertTweet actually set the

// hasGeotaggedStatuses bit for a user.

val setCounter = stats.counter("has\_geotag\_set")

def setHasGeotaggedStatuses(value: Boolean): FutureEffect[UserId] = {

val modifiedAccount = ModifiedAccount(hasGeotaggedStatuses = Some(value))

val modifiedUser = ModifiedUser(account = Some(modifiedAccount))

FutureEffect(userId => modifyAndGet((LookupContext(), userId, modifiedUser)).unit)

}

new GizmoduckUserGeotagUpdateStore {

override val asyncInsertTweet: FutureEffect[AsyncInsertTweet.Event] =

setHasGeotaggedStatuses(true)

.contramap[AsyncInsertTweet.Event](\_.user.id)

.onSuccess(\_ => setCounter.incr())

.onlyIf { e =>

// only with geo info and an account that doesn't yet have geotagged statuses flag set

hasGeo(e.tweet) && (e.user.account.exists(!\_.hasGeotaggedStatuses))

}

override val retryAsyncInsertTweet: FutureEffect[

TweetStoreRetryEvent[AsyncInsertTweet.Event]

] =

TweetStore.retry(Action, asyncInsertTweet)

override val scrubGeoUpdateUserTimestamp: FutureEffect[ScrubGeoUpdateUserTimestamp.Event] =

setHasGeotaggedStatuses(false)

.contramap[ScrubGeoUpdateUserTimestamp.Event](\_.userId)

.onlyIf(\_.mightHaveGeotaggedStatuses)

.onSuccess(\_ => clearedCounter.incr())

}

}

}