package com.twitter.tweetypie

package store

import com.twitter.tweetypie.backends.LimiterService

import com.twitter.tweetypie.thriftscala.\_

trait LimiterStore extends TweetStoreBase[LimiterStore] with InsertTweet.Store {

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

new TweetStoreWrapper(w, this) with LimiterStore with InsertTweet.StoreWrapper

}

object LimiterStore {

def apply(

incrementCreateSuccess: LimiterService.IncrementByOne,

incrementMediaTags: LimiterService.Increment

): LimiterStore =

new LimiterStore {

override val insertTweet: FutureEffect[InsertTweet.Event] =

FutureEffect[InsertTweet.Event] { event =>

Future.when(!event.dark) {

val userId = event.user.id

val contributorUserId: Option[UserId] = event.tweet.contributor.map(\_.userId)

val mediaTags = getMediaTagMap(event.tweet)

val mediaTagCount = countDistinctUserMediaTags(mediaTags)

Future

.join(

incrementCreateSuccess(userId, contributorUserId),

incrementMediaTags(userId, contributorUserId, mediaTagCount)

)

.unit

}

}

}

def countDistinctUserMediaTags(mediaTags: Map[MediaId, Seq[MediaTag]]): Int =

mediaTags.values.flatten.toSeq

.collect { case MediaTag(MediaTagType.User, Some(userId), \_, \_) => userId }

.distinct

.size

}