package com.twitter.tweetypie

package store

import com.twitter.servo.util.Scribe

import com.twitter.tweetypie.thriftscala.TweetMediaTagEvent

/\*\*

\* Scribes thrift-encoded TweetMediaTagEvents (from tweet\_events.thrift).

\*/

trait ScribeMediaTagStore extends TweetStoreBase[ScribeMediaTagStore] with AsyncInsertTweet.Store {

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

new TweetStoreWrapper(w, this) with ScribeMediaTagStore with AsyncInsertTweet.StoreWrapper

}

object ScribeMediaTagStore {

private def toMediaTagEvent(event: AsyncInsertTweet.Event): Option[TweetMediaTagEvent] = {

val tweet = event.tweet

val taggedUserIds = getMediaTagMap(tweet).values.flatten.flatMap(\_.userId).toSet

val timestamp = Time.now.inMilliseconds

if (taggedUserIds.nonEmpty) {

Some(TweetMediaTagEvent(tweet.id, getUserId(tweet), taggedUserIds, Some(timestamp)))

} else {

None

}

}

def apply(

scribe: FutureEffect[String] = Scribe("tweetypie\_media\_tag\_events")

): ScribeMediaTagStore =

new ScribeMediaTagStore {

override val asyncInsertTweet: FutureEffect[AsyncInsertTweet.Event] =

Scribe(TweetMediaTagEvent, scribe)

.contramapOption[AsyncInsertTweet.Event](toMediaTagEvent)

// we don't retry this action

override val retryAsyncInsertTweet: FutureEffect[

TweetStoreRetryEvent[AsyncInsertTweet.Event]

] =

FutureEffect.unit[TweetStoreRetryEvent[AsyncInsertTweet.Event]]

}

}