package com.twitter.tweetypie

package store

import com.twitter.tweetypie.tflock.TweetIndexer

import com.twitter.tweetypie.thriftscala.\_

trait TweetIndexingStore

extends TweetStoreBase[TweetIndexingStore]

with AsyncInsertTweet.Store

with AsyncDeleteTweet.Store

with AsyncUndeleteTweet.Store

with AsyncSetRetweetVisibility.Store {

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

new TweetStoreWrapper(w, this)

with TweetIndexingStore

with AsyncInsertTweet.StoreWrapper

with AsyncDeleteTweet.StoreWrapper

with AsyncUndeleteTweet.StoreWrapper

with AsyncSetRetweetVisibility.StoreWrapper

}

/\*\*

\* A TweetStore that sends indexing updates to a TweetIndexer.

\*/

object TweetIndexingStore {

val Action: AsyncWriteAction.TweetIndex.type = AsyncWriteAction.TweetIndex

def apply(indexer: TweetIndexer): TweetIndexingStore =

new TweetIndexingStore {

override val asyncInsertTweet: FutureEffect[AsyncInsertTweet.Event] =

FutureEffect[AsyncInsertTweet.Event](event => indexer.createIndex(event.tweet))

override val retryAsyncInsertTweet: FutureEffect[

TweetStoreRetryEvent[AsyncInsertTweet.Event]

] =

TweetStore.retry(Action, asyncInsertTweet)

override val asyncDeleteTweet: FutureEffect[AsyncDeleteTweet.Event] =

FutureEffect[AsyncDeleteTweet.Event](event =>

indexer.deleteIndex(event.tweet, event.isBounceDelete))

override val retryAsyncDeleteTweet: FutureEffect[

TweetStoreRetryEvent[AsyncDeleteTweet.Event]

] =

TweetStore.retry(Action, asyncDeleteTweet)

override val asyncUndeleteTweet: FutureEffect[AsyncUndeleteTweet.Event] =

FutureEffect[AsyncUndeleteTweet.Event](event => indexer.undeleteIndex(event.tweet))

override val retryAsyncUndeleteTweet: FutureEffect[

TweetStoreRetryEvent[AsyncUndeleteTweet.Event]

] =

TweetStore.retry(Action, asyncUndeleteTweet)

override val asyncSetRetweetVisibility: FutureEffect[AsyncSetRetweetVisibility.Event] =

FutureEffect[AsyncSetRetweetVisibility.Event] { event =>

indexer.setRetweetVisibility(event.retweetId, event.visible)

}

override val retryAsyncSetRetweetVisibility: FutureEffect[

TweetStoreRetryEvent[AsyncSetRetweetVisibility.Event]

] =

TweetStore.retry(Action, asyncSetRetweetVisibility)

}

}