package com.twitter.tweetypie

package store

import com.twitter.tweetypie.thriftscala.\_

/\*\*

\* AsyncEnqueueStore converts certains TweetStoreEvent types into their async-counterpart

\* events, and enqueues those to a deferredrpc-backed ThriftTweetService instance.

\*/

trait AsyncEnqueueStore

extends TweetStoreBase[AsyncEnqueueStore]

with InsertTweet.Store

with DeleteTweet.Store

with UndeleteTweet.Store

with IncrFavCount.Store

with IncrBookmarkCount.Store

with SetAdditionalFields.Store

with SetRetweetVisibility.Store

with Takedown.Store

with DeleteAdditionalFields.Store

with UpdatePossiblySensitiveTweet.Store {

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

new TweetStoreWrapper[AsyncEnqueueStore](w, this)

with AsyncEnqueueStore

with InsertTweet.StoreWrapper

with DeleteTweet.StoreWrapper

with UndeleteTweet.StoreWrapper

with IncrFavCount.StoreWrapper

with IncrBookmarkCount.StoreWrapper

with SetAdditionalFields.StoreWrapper

with SetRetweetVisibility.StoreWrapper

with Takedown.StoreWrapper

with DeleteAdditionalFields.StoreWrapper

with UpdatePossiblySensitiveTweet.StoreWrapper

}

object AsyncEnqueueStore {

def apply(

tweetService: ThriftTweetService,

scrubUserInAsyncInserts: User => User,

scrubSourceTweetInAsyncInserts: Tweet => Tweet,

scrubSourceUserInAsyncInserts: User => User

): AsyncEnqueueStore =

new AsyncEnqueueStore {

override val insertTweet: FutureEffect[InsertTweet.Event] =

FutureEffect[InsertTweet.Event] { e =>

tweetService.asyncInsert(

e.toAsyncRequest(

scrubUserInAsyncInserts,

scrubSourceTweetInAsyncInserts,

scrubSourceUserInAsyncInserts

)

)

}

override val deleteTweet: FutureEffect[DeleteTweet.Event] =

FutureEffect[DeleteTweet.Event] { e => tweetService.asyncDelete(e.toAsyncRequest) }

override val undeleteTweet: FutureEffect[UndeleteTweet.Event] =

FutureEffect[UndeleteTweet.Event] { e =>

tweetService.asyncUndeleteTweet(e.toAsyncUndeleteTweetRequest)

}

override val incrFavCount: FutureEffect[IncrFavCount.Event] =

FutureEffect[IncrFavCount.Event] { e => tweetService.asyncIncrFavCount(e.toAsyncRequest) }

override val incrBookmarkCount: FutureEffect[IncrBookmarkCount.Event] =

FutureEffect[IncrBookmarkCount.Event] { e =>

tweetService.asyncIncrBookmarkCount(e.toAsyncRequest)

}

override val setAdditionalFields: FutureEffect[SetAdditionalFields.Event] =

FutureEffect[SetAdditionalFields.Event] { e =>

tweetService.asyncSetAdditionalFields(e.toAsyncRequest)

}

override val setRetweetVisibility: FutureEffect[SetRetweetVisibility.Event] =

FutureEffect[SetRetweetVisibility.Event] { e =>

tweetService.asyncSetRetweetVisibility(e.toAsyncRequest)

}

override val deleteAdditionalFields: FutureEffect[DeleteAdditionalFields.Event] =

FutureEffect[DeleteAdditionalFields.Event] { e =>

tweetService.asyncDeleteAdditionalFields(e.toAsyncRequest)

}

override val updatePossiblySensitiveTweet: FutureEffect[UpdatePossiblySensitiveTweet.Event] =

FutureEffect[UpdatePossiblySensitiveTweet.Event] { e =>

tweetService.asyncUpdatePossiblySensitiveTweet(e.toAsyncRequest)

}

override val takedown: FutureEffect[Takedown.Event] =

FutureEffect[Takedown.Event] { e => tweetService.asyncTakedown(e.toAsyncRequest) }

}

}