package com.twitter.tweetypie

package handler

import com.twitter.stitch.NotFound

import com.twitter.stitch.Stitch

import com.twitter.tweetypie.repository.TweetQuery

import com.twitter.tweetypie.repository.TweetRepository

import com.twitter.tweetypie.repository.UserKey

import com.twitter.tweetypie.repository.UserQueryOptions

import com.twitter.tweetypie.repository.UserRepository

import com.twitter.tweetypie.repository.UserVisibility

import com.twitter.tweetypie.store.AsyncDeleteAdditionalFields

import com.twitter.tweetypie.store.DeleteAdditionalFields

import com.twitter.tweetypie.store.TweetStoreEventOrRetry

import com.twitter.tweetypie.thriftscala.AsyncDeleteAdditionalFieldsRequest

import com.twitter.tweetypie.thriftscala.DeleteAdditionalFieldsRequest

object DeleteAdditionalFieldsBuilder {

type Type = DeleteAdditionalFieldsRequest => Future[Seq[DeleteAdditionalFields.Event]]

val tweetQueryOptions = TweetQuery.Options(include = GetTweetsHandler.BaseInclude)

def apply(tweetRepo: TweetRepository.Type): Type = {

def getTweet(tweetId: TweetId) =

Stitch.run(

tweetRepo(tweetId, tweetQueryOptions)

.rescue(HandlerError.translateNotFoundToClientError(tweetId))

)

request => {

Future.collect(

request.tweetIds.map { tweetId =>

getTweet(tweetId).map { tweet =>

DeleteAdditionalFields.Event(

tweetId = tweetId,

fieldIds = request.fieldIds,

userId = getUserId(tweet),

timestamp = Time.now

)

}

}

)

}

}

}

object AsyncDeleteAdditionalFieldsBuilder {

type Type = AsyncDeleteAdditionalFieldsRequest => Future[

TweetStoreEventOrRetry[AsyncDeleteAdditionalFields.Event]

]

val userQueryOpts: UserQueryOptions = UserQueryOptions(Set(UserField.Safety), UserVisibility.All)

def apply(userRepo: UserRepository.Type): Type = {

def getUser(userId: UserId): Future[User] =

Stitch.run(

userRepo(UserKey.byId(userId), userQueryOpts)

.rescue { case NotFound => Stitch.exception(HandlerError.userNotFound(userId)) }

)

request =>

getUser(request.userId).map { user =>

AsyncDeleteAdditionalFields.Event.fromAsyncRequest(request, user)

}

}

}