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.AsyncSetAdditionalFields

import com.twitter.tweetypie.store.SetAdditionalFields

import com.twitter.tweetypie.store.TweetStoreEventOrRetry

import com.twitter.tweetypie.thriftscala.AsyncSetAdditionalFieldsRequest

import com.twitter.tweetypie.thriftscala.SetAdditionalFieldsRequest

object SetAdditionalFieldsBuilder {

type Type = SetAdditionalFieldsRequest => Future[SetAdditionalFields.Event]

val tweetOptions: TweetQuery.Options = TweetQuery.Options(include = GetTweetsHandler.BaseInclude)

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

def getTweet(tweetId: TweetId) =

Stitch.run(

tweetRepo(tweetId, tweetOptions)

.rescue(HandlerError.translateNotFoundToClientError(tweetId))

)

request => {

getTweet(request.additionalFields.id).map { tweet =>

SetAdditionalFields.Event(

additionalFields = request.additionalFields,

userId = getUserId(tweet),

timestamp = Time.now

)

}

}

}

}

object AsyncSetAdditionalFieldsBuilder {

type Type = AsyncSetAdditionalFieldsRequest => Future[

TweetStoreEventOrRetry[AsyncSetAdditionalFields.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 =>

AsyncSetAdditionalFields.Event.fromAsyncRequest(request, user)

}

}

}