package com.twitter.tweetypie

package handler

import com.twitter.stitch.Stitch

import com.twitter.tweetypie.Future

import com.twitter.tweetypie.core.FilteredState

import com.twitter.tweetypie.repository.TweetQuery

import com.twitter.tweetypie.repository.TweetRepository

import com.twitter.tweetypie.thriftscala.\_

import com.twitter.timelineservice.{thriftscala => tls}

import com.twitter.tweetypie.backends.TimelineService.GetPerspectives

object UnretweetHandler {

type Type = UnretweetRequest => Future[UnretweetResult]

def apply(

deleteTweets: TweetDeletePathHandler.DeleteTweets,

getPerspectives: GetPerspectives,

unretweetEdits: TweetDeletePathHandler.UnretweetEdits,

tweetRepo: TweetRepository.Type,

): Type = { request: UnretweetRequest =>

val handleEdits = getSourceTweet(request.sourceTweetId, tweetRepo).liftToTry.flatMap {

case Return(sourceTweet) =>

// If we're able to fetch the source Tweet, unretweet all its other versions

unretweetEdits(sourceTweet.editControl, request.sourceTweetId, request.userId)

case Throw(\_) => Future.Done

}

handleEdits.flatMap(\_ => unretweetSourceTweet(request, deleteTweets, getPerspectives))

}

def unretweetSourceTweet(

request: UnretweetRequest,

deleteTweets: TweetDeletePathHandler.DeleteTweets,

getPerspectives: GetPerspectives,

): Future[UnretweetResult] =

getPerspectives(

Seq(tls.PerspectiveQuery(request.userId, Seq(request.sourceTweetId)))

).map { results => results.head.perspectives.headOption.flatMap(\_.retweetId) }

.flatMap {

case Some(id) =>

deleteTweets(

DeleteTweetsRequest(tweetIds = Seq(id), byUserId = Some(request.userId)),

false

).map(\_.head).map { deleteTweetResult =>

UnretweetResult(Some(deleteTweetResult.tweetId), deleteTweetResult.state)

}

case None => Future.value(UnretweetResult(None, TweetDeleteState.Ok))

}

def getSourceTweet(

sourceTweetId: TweetId,

tweetRepo: TweetRepository.Type

): Future[Tweet] = {

val options: TweetQuery.Options = TweetQuery

.Options(include = TweetQuery.Include(tweetFields = Set(Tweet.EditControlField.id)))

Stitch.run {

tweetRepo(sourceTweetId, options).rescue {

case \_: FilteredState => Stitch.NotFound

}

}

}

}