package com.twitter.tweetypie

package hydrator

import com.twitter.stitch.NotFound

import com.twitter.stitch.Stitch

import com.twitter.tweetypie.core.FilteredState.Unavailable.\_

import com.twitter.tweetypie.core.TweetResult

import com.twitter.tweetypie.core.ValueState

import com.twitter.tweetypie.repository.TweetQuery

import com.twitter.tweetypie.repository.TweetResultRepository

import com.twitter.tweetypie.thriftscala.DetachedRetweet

/\*\*

\* Loads the source tweet for a retweet

\*/

object SourceTweetHydrator {

type Type = ValueHydrator[Option[TweetResult], TweetCtx]

def configureOptions(opts: TweetQuery.Options): TweetQuery.Options = {

// set scrubUnrequestedFields to false so that we will have access to

// additional fields, which will be copied into the retweet.

// set fetchStoredTweets to false because we don't want to fetch and hydrate

// the source tweet if it is deleted.

opts.copy(scrubUnrequestedFields = false, fetchStoredTweets = false, isSourceTweet = true)

}

private object NotFoundException {

def unapply(t: Throwable): Option[Boolean] =

t match {

case NotFound => Some(false)

case TweetDeleted | BounceDeleted => Some(true)

case \_ => None

}

}

def apply(

repo: TweetResultRepository.Type,

stats: StatsReceiver,

scribeDetachedRetweets: FutureEffect[DetachedRetweet] = FutureEffect.unit

): Type = {

val notFoundCounter = stats.counter("not\_found")

ValueHydrator[Option[TweetResult], TweetCtx] { (\_, ctx) =>

ctx.sourceTweetId match {

case None =>

ValueState.StitchUnmodifiedNone

case Some(srcTweetId) =>

repo(srcTweetId, configureOptions(ctx.opts)).liftToTry.flatMap {

case Throw(NotFoundException(isDeleted)) =>

notFoundCounter.incr()

scribeDetachedRetweets(detachedRetweet(srcTweetId, ctx))

if (ctx.opts.requireSourceTweet) {

Stitch.exception(SourceTweetNotFound(isDeleted))

} else {

ValueState.StitchUnmodifiedNone

}

case Return(r) => Stitch.value(ValueState.modified(Some(r)))

case Throw(t) => Stitch.exception(t)

}

}

}.onlyIf((curr, \_) => curr.isEmpty)

}

def detachedRetweet(srcTweetId: TweetId, ctx: TweetCtx): DetachedRetweet =

DetachedRetweet(ctx.tweetId, ctx.userId, srcTweetId)

}