package com.twitter.tweetypie

package hydrator

import com.twitter.stitch.Stitch

import com.twitter.tweetypie.core.\_

import com.twitter.tweetypie.repository.\_

/\*\*

\* Loads the tweet referenced by `Tweet.quotedTweet`.

\*/

object QuotedTweetHydrator {

type Type = ValueHydrator[Option[QuotedTweetResult], Ctx]

case class Ctx(

quotedTweetFilteredState: Option[FilteredState.Unavailable],

underlyingTweetCtx: TweetCtx)

extends TweetCtx.Proxy

def apply(repo: TweetResultRepository.Type): Type = {

ValueHydrator[Option[QuotedTweetResult], Ctx] { (\_, ctx) =>

(ctx.quotedTweetFilteredState, ctx.quotedTweet) match {

case (\_, None) =>

// If there is no quoted tweet ref, leave the value as None,

// indicating undefined

ValueState.StitchUnmodifiedNone

case (Some(fs), \_) =>

Stitch.value(ValueState.modified(Some(QuotedTweetResult.Filtered(fs))))

case (None, Some(qtRef)) =>

val qtQueryOptions =

ctx.opts.copy(

// we don't want to recursively load quoted tweets

include = ctx.opts.include.copy(quotedTweet = false),

// be sure to get a clean version of the tweet

scrubUnrequestedFields = true,

// TweetVisibilityLibrary filters quoted tweets slightly differently from other tweets.

// Specifically, most Interstitial verdicts are converted to Drops.

isInnerQuotedTweet = true

)

repo(qtRef.tweetId, qtQueryOptions).transform { t =>

Stitch.const {

QuotedTweetResult.fromTry(t).map(r => ValueState.modified(Some(r)))

}

}

}

}.onlyIf((curr, ctx) => curr.isEmpty && ctx.opts.include.quotedTweet)

}

}