package com.twitter.tweetypie

package repository

import com.twitter.spam.rtf.thriftscala.SafetyLevel

import com.twitter.stitch.NotFound

import com.twitter.stitch.Stitch

import com.twitter.tweetypie.core.FilteredState.Unavailable.TweetDeleted

import com.twitter.tweetypie.thriftscala.ConversationControl

/\*\*

\* This repository loads up the conversation control values for a tweet which controls who can reply

\* to a tweet. Because the conversation control values are stored on the root tweet of a conversation,

\* we need to make sure that the code is able to load the data from the root tweet. To ensure this,

\* no visibility filtering options are set on the query to load the root tweet fields.

\*

\* If visibility filtering was enabled, and the root tweet was filtered for the requesting user,

\* then the conversation control data would not be returned and enforcement would effectively be

\* side-stepped.

\*/

object ConversationControlRepository {

private[this] val log = Logger(getClass)

type Type = (TweetId, CacheControl) => Stitch[Option[ConversationControl]]

def apply(repo: TweetRepository.Type, stats: StatsReceiver): Type =

(conversationId: TweetId, cacheControl: CacheControl) => {

val options = TweetQuery.Options(

include = TweetQuery.Include(Set(Tweet.ConversationControlField.id)),

// We want the root tweet of a conversation that we're looking up to be

// cached with the same policy as the tweet we're looking up.

cacheControl = cacheControl,

enforceVisibilityFiltering = false,

safetyLevel = SafetyLevel.FilterNone

)

repo(conversationId, options)

.map(rootTweet => rootTweet.conversationControl)

.handle {

// We don't know of any case where tweets would return NotFound, but for

// for pragmatic reasons, we're opening the conversation for replies

// in case a bug causing tweets to be NotFound exists.

case NotFound =>

stats.counter("tweet\_not\_found")

None

// If no root tweet is found, the reply has no conversation controls

// this is by design, deleting the root tweet "opens" the conversation

case TweetDeleted =>

stats.counter("tweet\_deleted")

None

}

}

}