package com.twitter.tweetypie

package hydrator

import com.twitter.stitch.Stitch

import com.twitter.tweetypie.core.\_

import com.twitter.tweetypie.repository.\_

import com.twitter.tweetypie.util.CommunityUtil

object TweetVisibilityHydrator {

type Type = ValueHydrator[Option[FilteredState.Suppress], Ctx]

case class Ctx(tweet: Tweet, underlyingTweetCtx: TweetCtx) extends TweetCtx.Proxy

def apply(

repo: TweetVisibilityRepository.Type,

failClosedInVF: Gate[Unit],

stats: StatsReceiver

): Type = {

val outcomeScope = stats.scope("outcome")

val unavailable = outcomeScope.counter("unavailable")

val suppress = outcomeScope.counter("suppress")

val allow = outcomeScope.counter("allow")

val failClosed = outcomeScope.counter("fail\_closed")

val communityFailClosed = outcomeScope.counter("community\_fail\_closed")

val failOpen = outcomeScope.counter("fail\_open")

ValueHydrator[Option[FilteredState.Suppress], Ctx] { (curr, ctx) =>

val request = TweetVisibilityRepository.Request(

tweet = ctx.tweet,

viewerId = ctx.opts.forUserId,

safetyLevel = ctx.opts.safetyLevel,

isInnerQuotedTweet = ctx.opts.isInnerQuotedTweet,

isRetweet = ctx.isRetweet,

hydrateConversationControl = ctx.tweetFieldRequested(Tweet.ConversationControlField),

isSourceTweet = ctx.opts.isSourceTweet

)

repo(request).liftToTry.flatMap {

// If FilteredState.Unavailable is returned from repo then throw it

case Return(Some(fs: FilteredState.Unavailable)) =>

unavailable.incr()

Stitch.exception(fs)

// If FilteredState.Suppress is returned from repo then return it

case Return(Some(fs: FilteredState.Suppress)) =>

suppress.incr()

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

// If None is returned from repo then return unmodified

case Return(None) =>

allow.incr()

ValueState.StitchUnmodifiedNone

// Propagate thrown exceptions if fail closed

case Throw(e) if failClosedInVF() =>

failClosed.incr()

Stitch.exception(e)

// Community tweets are special cased to fail closed to avoid

// leaking tweets expected to be private to a community.

case Throw(e) if CommunityUtil.hasCommunity(request.tweet.communities) =>

communityFailClosed.incr()

Stitch.exception(e)

case Throw(\_) =>

failOpen.incr()

Stitch.value(ValueState.unmodified(curr))

}

}

}

}