package com.twitter.tweetypie.handler

import com.twitter.stitch.Stitch

import com.twitter.tweetypie.core.TweetCreateFailure

import com.twitter.tweetypie.repository.ConversationControlRepository

import com.twitter.tweetypie.repository.TweetQuery

import com.twitter.tweetypie.thriftscala.ExclusiveTweetControl

import com.twitter.tweetypie.thriftscala.ExclusiveTweetControlOptions

import com.twitter.tweetypie.thriftscala.QuotedTweet

import com.twitter.tweetypie.thriftscala.TrustedFriendsControl

import com.twitter.tweetypie.thriftscala.TrustedFriendsControlOptions

import com.twitter.tweetypie.thriftscala.TweetCreateState

import com.twitter.tweetypie.FutureEffect

import com.twitter.tweetypie.Gate

import com.twitter.tweetypie.TweetId

import com.twitter.tweetypie.UserId

import com.twitter.tweetypie.thriftscala.EditControl

import com.twitter.tweetypie.thriftscala.EditOptions

import com.twitter.visibility.writer.interfaces.tweets.TweetWriteEnforcementLibrary

import com.twitter.visibility.writer.interfaces.tweets.TweetWriteEnforcementRequest

import com.twitter.visibility.writer.models.ActorContext

import com.twitter.visibility.writer.Allow

import com.twitter.visibility.writer.Deny

import com.twitter.visibility.writer.DenyExclusiveTweetReply

import com.twitter.visibility.writer.DenyStaleTweetQuoteTweet

import com.twitter.visibility.writer.DenyStaleTweetReply

import com.twitter.visibility.writer.DenySuperFollowsCreate

import com.twitter.visibility.writer.DenyTrustedFriendsCreate

import com.twitter.visibility.writer.DenyTrustedFriendsQuoteTweet

import com.twitter.visibility.writer.DenyTrustedFriendsReply

object TweetWriteValidator {

case class Request(

conversationId: Option[TweetId],

userId: UserId,

exclusiveTweetControlOptions: Option[ExclusiveTweetControlOptions],

replyToExclusiveTweetControl: Option[ExclusiveTweetControl],

trustedFriendsControlOptions: Option[TrustedFriendsControlOptions],

inReplyToTrustedFriendsControl: Option[TrustedFriendsControl],

quotedTweetOpt: Option[QuotedTweet],

inReplyToTweetId: Option[TweetId],

inReplyToEditControl: Option[EditControl],

editOptions: Option[EditOptions])

type Type = FutureEffect[Request]

def apply(

convoCtlRepo: ConversationControlRepository.Type,

tweetWriteEnforcementLibrary: TweetWriteEnforcementLibrary,

enableExclusiveTweetControlValidation: Gate[Unit],

enableTrustedFriendsControlValidation: Gate[Unit],

enableStaleTweetValidation: Gate[Unit]

): FutureEffect[Request] =

FutureEffect[Request] { request =>

// We are creating up an empty TweetQuery.Options here so we can use the default

// CacheControl value and avoid hard coding it here.

val queryOptions = TweetQuery.Options(TweetQuery.Include())

Stitch.run {

for {

convoCtl <- request.conversationId match {

case Some(convoId) =>

convoCtlRepo(

convoId,

queryOptions.cacheControl

)

case None =>

Stitch.value(None)

}

result <- tweetWriteEnforcementLibrary(

TweetWriteEnforcementRequest(

rootConversationControl = convoCtl,

convoId = request.conversationId,

exclusiveTweetControlOptions = request.exclusiveTweetControlOptions,

replyToExclusiveTweetControl = request.replyToExclusiveTweetControl,

trustedFriendsControlOptions = request.trustedFriendsControlOptions,

inReplyToTrustedFriendsControl = request.inReplyToTrustedFriendsControl,

quotedTweetOpt = request.quotedTweetOpt,

actorContext = ActorContext(request.userId),

inReplyToTweetId = request.inReplyToTweetId,

inReplyToEditControl = request.inReplyToEditControl,

editOptions = request.editOptions

),

enableExclusiveTweetControlValidation = enableExclusiveTweetControlValidation,

enableTrustedFriendsControlValidation = enableTrustedFriendsControlValidation,

enableStaleTweetValidation = enableStaleTweetValidation

)

\_ <- result match {

case Allow =>

Stitch.Done

case Deny =>

Stitch.exception(TweetCreateFailure.State(TweetCreateState.ReplyTweetNotAllowed))

case DenyExclusiveTweetReply =>

Stitch.exception(

TweetCreateFailure.State(TweetCreateState.ExclusiveTweetEngagementNotAllowed))

case DenySuperFollowsCreate =>

Stitch.exception(

TweetCreateFailure.State(TweetCreateState.SuperFollowsCreateNotAuthorized))

case DenyTrustedFriendsReply =>

Stitch.exception(

TweetCreateFailure.State(TweetCreateState.TrustedFriendsEngagementNotAllowed))

case DenyTrustedFriendsCreate =>

Stitch.exception(

TweetCreateFailure.State(TweetCreateState.TrustedFriendsCreateNotAllowed))

case DenyTrustedFriendsQuoteTweet =>

Stitch.exception(

TweetCreateFailure.State(TweetCreateState.TrustedFriendsQuoteTweetNotAllowed))

case DenyStaleTweetReply =>

Stitch.exception(

TweetCreateFailure.State(TweetCreateState.StaleTweetEngagementNotAllowed))

case DenyStaleTweetQuoteTweet =>

Stitch.exception(

TweetCreateFailure.State(TweetCreateState.StaleTweetQuoteTweetNotAllowed))

}

} yield ()

}

}

}