package com.twitter.tweetypie.handler

import com.twitter.gizmoduck.thriftscala.User

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

import com.twitter.tweetypie.repository.CacheControl

import com.twitter.tweetypie.repository.TweetQuery

import com.twitter.tweetypie.thriftscala.MediaEntity

import com.twitter.tweetypie.thriftscala.StatusCounts

import com.twitter.tweetypie.thriftscala.Tweet

import com.twitter.tweetypie.thriftscala.WritePathHydrationOptions

object WritePathQueryOptions {

/\*\*

\* Base TweetQuery.Include for all hydration options.

\*/

val BaseInclude: TweetQuery.Include =

GetTweetsHandler.BaseInclude.also(

tweetFields = Set(

Tweet.CardReferenceField.id,

Tweet.MediaTagsField.id,

Tweet.SelfPermalinkField.id,

Tweet.ExtendedTweetMetadataField.id,

Tweet.VisibleTextRangeField.id,

Tweet.NsfaHighRecallLabelField.id,

Tweet.CommunitiesField.id,

Tweet.ExclusiveTweetControlField.id,

Tweet.TrustedFriendsControlField.id,

Tweet.CollabControlField.id,

Tweet.EditControlField.id,

Tweet.EditPerspectiveField.id,

Tweet.NoteTweetField.id

)

)

/\*\*

\* Base TweetQuery.Include for all creation-related hydrations.

\*/

val BaseCreateInclude: TweetQuery.Include =

BaseInclude

.also(

tweetFields = Set(

Tweet.PlaceField.id,

Tweet.ProfileGeoEnrichmentField.id,

Tweet.SelfThreadMetadataField.id

),

mediaFields = Set(MediaEntity.AdditionalMetadataField.id),

quotedTweet = Some(true),

pastedMedia = Some(true)

)

/\*\*

\* Base TweetQuery.Include for all deletion-related hydrations.

\*/

val BaseDeleteInclude: TweetQuery.Include = BaseInclude

.also(tweetFields =

Set(Tweet.BounceLabelField.id, Tweet.ConversationControlField.id, Tweet.EditControlField.id))

val AllCounts: Set[Short] = StatusCounts.fieldInfos.map(\_.tfield.id).toSet

def insert(

cause: TweetQuery.Cause,

user: User,

options: WritePathHydrationOptions,

isEditControlEdit: Boolean

): TweetQuery.Options =

createOptions(

writePathHydrationOptions = options,

includePerspective = false,

// include counts if tweet edit, otherwise false

includeCounts = isEditControlEdit,

cause = cause,

forUser = user,

// Do not perform any filtering when we are hydrating the tweet we are creating

safetyLevel = SafetyLevel.FilterNone

)

def retweetSourceTweet(user: User, options: WritePathHydrationOptions): TweetQuery.Options =

createOptions(

writePathHydrationOptions = options,

includePerspective = true,

includeCounts = true,

cause = TweetQuery.Cause.Read,

forUser = user,

// If Scarecrow is down, we may proceed with creating a RT. The safetyLevel is necessary

// to prevent so that the inner tweet's count is not sent in the TweetCreateEvent we send

// to EventBus. If this were emitted, live pipeline would publish counts to the clients.

safetyLevel = SafetyLevel.TweetWritesApi

)

def quotedTweet(user: User, options: WritePathHydrationOptions): TweetQuery.Options =

createOptions(

writePathHydrationOptions = options,

includePerspective = true,

includeCounts = true,

cause = TweetQuery.Cause.Read,

forUser = user,

// We pass in the safetyLevel so that the inner tweet's are excluded

// from the TweetCreateEvent we send to EventBus. If this were emitted,

// live pipeline would publish counts to the clients.

safetyLevel = SafetyLevel.TweetWritesApi

)

private def condSet[A](cond: Boolean, item: A): Set[A] =

if (cond) Set(item) else Set.empty

private def createOptions(

writePathHydrationOptions: WritePathHydrationOptions,

includePerspective: Boolean,

includeCounts: Boolean,

cause: TweetQuery.Cause,

forUser: User,

safetyLevel: SafetyLevel,

): TweetQuery.Options = {

val cardsEnabled: Boolean = writePathHydrationOptions.includeCards

val cardsPlatformKeySpecified: Boolean = writePathHydrationOptions.cardsPlatformKey.nonEmpty

val cardsV1Enabled: Boolean = cardsEnabled && !cardsPlatformKeySpecified

val cardsV2Enabled: Boolean = cardsEnabled && cardsPlatformKeySpecified

TweetQuery.Options(

include = BaseCreateInclude.also(

tweetFields =

condSet(includePerspective, Tweet.PerspectiveField.id) ++

condSet(cardsV1Enabled, Tweet.CardsField.id) ++

condSet(cardsV2Enabled, Tweet.Card2Field.id) ++

condSet(includeCounts, Tweet.CountsField.id) ++

// for PreviousCountsField, copy includeCounts state on the write path

condSet(includeCounts, Tweet.PreviousCountsField.id) ++

// hydrate ConversationControl on Reply Tweet creations so clients can consume

Set(Tweet.ConversationControlField.id),

countsFields = if (includeCounts) AllCounts else Set.empty

),

cause = cause,

forUserId = Some(forUser.id),

cardsPlatformKey = writePathHydrationOptions.cardsPlatformKey,

languageTag = forUser.account.map(\_.language).getOrElse("en"),

extensionsArgs = writePathHydrationOptions.extensionsArgs,

safetyLevel = safetyLevel,

simpleQuotedTweet = writePathHydrationOptions.simpleQuotedTweet

)

}

def deleteTweets: TweetQuery.Options =

TweetQuery.Options(

include = BaseDeleteInclude,

cacheControl = CacheControl.ReadOnlyCache,

extensionsArgs = None,

requireSourceTweet = false // retweet should be deletable even if source tweet missing

)

def deleteTweetsWithoutEditControl: TweetQuery.Options =

deleteTweets.copy(enableEditControlHydration = false)

}