package com.twitter.tweetypie

package hydrator

import com.twitter.mediaservices.commons.thriftscala.MediaKey

import com.twitter.mediaservices.media\_util.GenericMediaKey

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

import com.twitter.stitch.Stitch

import com.twitter.tweetypie.core.ValueState

import com.twitter.tweetypie.thriftscala.MediaEntity

import com.twitter.tweetypie.thriftscala.UrlEntity

import com.twitter.tweetypie.media.thriftscala.MediaRef

import com.twitter.tweetypie.repository.TweetQuery

import com.twitter.tweetypie.repository.TweetRepository

import com.twitter.tweetypie.thriftscala.FieldByPath

/\*\*

\* MediaRefsHydrator hydrates the Tweet.mediaRefs field based on stored media keys

\* and pasted media. Media keys are available in three ways:

\*

\* 1. (For old Tweets): in the stored MediaEntity

\* 2. (For 2016+ Tweets): in the mediaKeys field

\* 3. From other Tweets using pasted media

\*

\* This hydrator combines these three sources into a single field, providing the

\* media key and source Tweet information for pasted media.

\*

\* Long-term we will move this logic to the write path and backfill the field for old Tweets.

\*/

object MediaRefsHydrator {

type Type = ValueHydrator[Option[Seq[MediaRef]], Ctx]

case class Ctx(

media: Seq[MediaEntity],

mediaKeys: Seq[MediaKey],

urlEntities: Seq[UrlEntity],

underlyingTweetCtx: TweetCtx)

extends TweetCtx.Proxy {

def includePastedMedia: Boolean = opts.include.pastedMedia

}

val hydratedField: FieldByPath = fieldByPath(Tweet.MediaRefsField)

def mediaKeyToMediaRef(mediaKey: MediaKey): MediaRef =

MediaRef(

genericMediaKey = GenericMediaKey(mediaKey).toStringKey()

)

// Convert a pasted Tweet into a Seq of MediaRef from that Tweet with the correct sourceTweetId and sourceUserId

def pastedTweetToMediaRefs(

tweet: Tweet

): Seq[MediaRef] =

tweet.mediaRefs.toSeq.flatMap { mediaRefs =>

mediaRefs.map(

\_.copy(

sourceTweetId = Some(tweet.id),

sourceUserId = Some(getUserId(tweet))

))

}

// Fetch MediaRefs from pasted media Tweet URLs in the Tweet text

def getPastedMediaRefs(

repo: TweetRepository.Optional,

ctx: Ctx,

includePastedMedia: Gate[Unit]

): Stitch[Seq[MediaRef]] = {

if (includePastedMedia() && ctx.includePastedMedia) {

// Extract Tweet ids from pasted media permalinks in the Tweet text

val pastedMediaTweetIds: Seq[TweetId] =

PastedMediaHydrator.pastedIdsAndEntities(ctx.tweetId, ctx.urlEntities).map(\_.\_1)

val opts = TweetQuery.Options(

include = TweetQuery.Include(

tweetFields = Set(Tweet.CoreDataField.id, Tweet.MediaRefsField.id),

pastedMedia = false // don't recursively load pasted media refs

))

// Load a Seq of Tweets with pasted media, ignoring any returned with NotFound or a FilteredState

val pastedTweets: Stitch[Seq[Tweet]] = Stitch

.traverse(pastedMediaTweetIds) { id =>

repo(id, opts)

}.map(\_.flatten)

pastedTweets.map(\_.flatMap(pastedTweetToMediaRefs))

} else {

Stitch.Nil

}

}

// Make empty Seq None and non-empty Seq Some(Seq(...)) to comply with the thrift field type

def optionalizeSeq(mediaRefs: Seq[MediaRef]): Option[Seq[MediaRef]] =

Some(mediaRefs).filterNot(\_.isEmpty)

def apply(

repo: TweetRepository.Optional,

includePastedMedia: Gate[Unit]

): Type = {

ValueHydrator[Option[Seq[MediaRef]], Ctx] { (curr, ctx) =>

// Fetch mediaRefs from Tweet media

val storedMediaRefs: Seq[MediaRef] = ctx.media.map { mediaEntity =>

// Use MediaKeyHydrator.infer to determine the media key from the media entity

val mediaKey = MediaKeyHydrator.infer(Some(ctx.mediaKeys), mediaEntity)

mediaKeyToMediaRef(mediaKey)

}

// Fetch mediaRefs from pasted media

getPastedMediaRefs(repo, ctx, includePastedMedia).liftToTry.map {

case Return(pastedMediaRefs) =>

// Combine the refs from the Tweet's own media and those from pasted media, then limit

// to MaxMediaEntitiesPerTweet.

val limitedRefs =

(storedMediaRefs ++ pastedMediaRefs).take(PastedMediaHydrator.MaxMediaEntitiesPerTweet)

ValueState.delta(curr, optionalizeSeq(limitedRefs))

case Throw(\_) =>

ValueState.partial(optionalizeSeq(storedMediaRefs), hydratedField)

}

}.onlyIf { (\_, ctx) =>

ctx.tweetFieldRequested(Tweet.MediaRefsField) ||

ctx.opts.safetyLevel != SafetyLevel.FilterNone

}

}

}