package com.twitter.tweetypie

package handler

import com.twitter.servo.util.FutureArrow

import com.twitter.takedown.util.TakedownReasons.\_

import com.twitter.tweetypie.store.Takedown

import com.twitter.tweetypie.thriftscala.TakedownRequest

import com.twitter.tweetypie.thriftscala.Tweet

import com.twitter.tweetypie.util.Takedowns

/\*\*

\* This handler processes TakedownRequest objects sent to Tweetypie's takedown endpoint.

\* The request object specifies which takedown countries are being added and which are

\* being removed. It also includes side effect flags for setting the tweet's has\_takedown

\* bit, scribing to Guano, and enqueuing to EventBus. For more information about inputs

\* to the takedown endpoint, see the TakedownRequest documentation in the thrift definition.

\*/

object TakedownHandler {

type Type = FutureArrow[TakedownRequest, Unit]

def apply(

getTweet: FutureArrow[TweetId, Tweet],

getUser: FutureArrow[UserId, User],

writeTakedown: FutureEffect[Takedown.Event]

): Type = {

FutureArrow { request =>

for {

tweet <- getTweet(request.tweetId)

user <- getUser(getUserId(tweet))

userHasTakedowns = user.takedowns.map(userTakedownsToReasons).exists(\_.nonEmpty)

existingTweetReasons = Takedowns.fromTweet(tweet).reasons

reasonsToRemove = (request.countriesToRemove.map(countryCodeToReason) ++

request.reasonsToRemove.map(normalizeReason)).distinct.sortBy(\_.toString)

reasonsToAdd = (request.countriesToAdd.map(countryCodeToReason) ++

request.reasonsToAdd.map(normalizeReason)).distinct.sortBy(\_.toString)

updatedTweetTakedowns =

(existingTweetReasons ++ reasonsToAdd)

.filterNot(reasonsToRemove.contains)

.toSeq

.sortBy(\_.toString)

(cs, rs) = Takedowns.partitionReasons(updatedTweetTakedowns)

updatedTweet = Lens.setAll(

tweet,

// these fields are cached on the Tweet in CachingTweetStore and written in

// ManhattanTweetStore

TweetLenses.hasTakedown -> (updatedTweetTakedowns.nonEmpty || userHasTakedowns),

TweetLenses.tweetypieOnlyTakedownCountryCodes -> Some(cs).filter(\_.nonEmpty),

TweetLenses.tweetypieOnlyTakedownReasons -> Some(rs).filter(\_.nonEmpty)

)

\_ <- writeTakedown.when(tweet != updatedTweet) {

Takedown.Event(

tweet = updatedTweet,

timestamp = Time.now,

user = Some(user),

takedownReasons = updatedTweetTakedowns,

reasonsToAdd = reasonsToAdd,

reasonsToRemove = reasonsToRemove,

auditNote = request.auditNote,

host = request.host,

byUserId = request.byUserId,

eventbusEnqueue = request.eventbusEnqueue,

scribeForAudit = request.scribeForAudit,

updateCodesAndReasons = true

)

}

} yield ()

}

}

}