package com.twitter.tweetypie.storage

import com.twitter.tweetypie.storage\_internal.thriftscala.\_

import com.twitter.tbird.{thriftscala => tbird}

object StatusConversions {

/\*\*

\* This is used only in Scribe.scala, when scribing to tbird\_add\_status

\* Once we remove that, we can also remove this.

\*/

def toTBirdStatus(tweet: StoredTweet): tbird.Status =

tbird.Status(

id = tweet.id,

userId = tweet.userId.get,

text = tweet.text.get,

createdVia = tweet.createdVia.get,

createdAtSec = tweet.createdAtSec.get,

reply = tweet.reply.map(toTBirdReply),

share = tweet.share.map(toTBirdShare),

contributorId = tweet.contributorId,

geo = tweet.geo.map(toTBirdGeo),

hasTakedown = tweet.hasTakedown.getOrElse(false),

nsfwUser = tweet.nsfwUser.getOrElse(false),

nsfwAdmin = tweet.nsfwAdmin.getOrElse(false),

media = tweet.media.map(\_.map(toTBirdMedia)).getOrElse(Seq()),

narrowcast = tweet.narrowcast.map(toTBirdNarrowcast),

nullcast = tweet.nullcast.getOrElse(false),

trackingId = tweet.trackingId

)

/\*\*

\* This is only used in a test, to verify that the above method `toTBirdStatus`

\* works, so we can't remove it as long as the above method exists.

\*/

def fromTBirdStatus(status: tbird.Status): StoredTweet = {

StoredTweet(

id = status.id,

userId = Some(status.userId),

text = Some(status.text),

createdVia = Some(status.createdVia),

createdAtSec = Some(status.createdAtSec),

reply = status.reply.map(fromTBirdReply),

share = status.share.map(fromTBirdShare),

contributorId = status.contributorId,

geo = status.geo.map(fromTBirdGeo),

hasTakedown = Some(status.hasTakedown),

nsfwUser = Some(status.nsfwUser),

nsfwAdmin = Some(status.nsfwAdmin),

media = Some(status.media.map(fromTBirdMedia)),

narrowcast = status.narrowcast.map(fromTBirdNarrowcast),

nullcast = Some(status.nullcast),

trackingId = status.trackingId

)

}

private def fromTBirdReply(reply: tbird.Reply): StoredReply =

StoredReply(

inReplyToStatusId = reply.inReplyToStatusId,

inReplyToUserId = reply.inReplyToUserId

)

private def fromTBirdShare(share: tbird.Share): StoredShare =

StoredShare(

sourceStatusId = share.sourceStatusId,

sourceUserId = share.sourceUserId,

parentStatusId = share.parentStatusId

)

private def fromTBirdGeo(geo: tbird.Geo): StoredGeo =

StoredGeo(

latitude = geo.latitude,

longitude = geo.longitude,

geoPrecision = geo.geoPrecision,

entityId = geo.entityId

)

private def fromTBirdMedia(media: tbird.MediaEntity): StoredMediaEntity =

StoredMediaEntity(

id = media.id,

mediaType = media.mediaType,

width = media.width,

height = media.height

)

private def fromTBirdNarrowcast(narrowcast: tbird.Narrowcast): StoredNarrowcast =

StoredNarrowcast(

language = Some(narrowcast.language),

location = Some(narrowcast.location),

ids = Some(narrowcast.ids)

)

private def toTBirdReply(reply: StoredReply): tbird.Reply =

tbird.Reply(

inReplyToStatusId = reply.inReplyToStatusId,

inReplyToUserId = reply.inReplyToUserId

)

private def toTBirdShare(share: StoredShare): tbird.Share =

tbird.Share(

sourceStatusId = share.sourceStatusId,

sourceUserId = share.sourceUserId,

parentStatusId = share.parentStatusId

)

private def toTBirdGeo(geo: StoredGeo): tbird.Geo =

tbird.Geo(

latitude = geo.latitude,

longitude = geo.longitude,

geoPrecision = geo.geoPrecision,

entityId = geo.entityId,

name = geo.name

)

private def toTBirdMedia(media: StoredMediaEntity): tbird.MediaEntity =

tbird.MediaEntity(

id = media.id,

mediaType = media.mediaType,

width = media.width,

height = media.height

)

private def toTBirdNarrowcast(narrowcast: StoredNarrowcast): tbird.Narrowcast =

tbird.Narrowcast(

language = narrowcast.language.getOrElse(Nil),

location = narrowcast.location.getOrElse(Nil),

ids = narrowcast.ids.getOrElse(Nil)

)

}