package com.twitter.tweetypie

package hydrator

import com.twitter.mediaservices.commons.tweetmedia.thriftscala.\_

import com.twitter.tweetypie.media.\_

import com.twitter.tweetypie.thriftscala.\_

import scala.collection.Set

/\*\*

\* Removes partial Url, Media, and Mention entities that were not

\* fully hydrated. Rather than returning no value or a value with

\* incomplete entities on an entity hydration failure, we gracefully

\* degrade to just omitting those entities. This step needs to be

\* applied in the post-cache filter, so that we don't cache the value

\* with missing entities.

\*

\* A MediaEntity will first be converted back to a UrlEntity if it is only

\* partially hydrated. If the resulting UrlEntity is itself then only partially

\* hydrated, it will get dropped also.

\*/

object PartialEntityCleaner {

def apply(stats: StatsReceiver): Mutation[Tweet] = {

val scopedStats = stats.scope("partial\_entity\_cleaner")

Mutation

.all(

Seq(

TweetLenses.urls.mutation(urls.countMutations(scopedStats.counter("urls"))),

TweetLenses.media.mutation(media.countMutations(scopedStats.counter("media"))),

TweetLenses.mentions.mutation(mentions.countMutations(scopedStats.counter("mentions")))

)

)

.onlyIf(!isRetweet(\_))

}

private[this] def clean[E](isPartial: E => Boolean) =

Mutation[Seq[E]] { items =>

items.partition(isPartial) match {

case (Nil, nonPartial) => None

case (partial, nonPartial) => Some(nonPartial)

}

}

private[this] val mentions =

clean[MentionEntity](e => e.userId.isEmpty || e.name.isEmpty)

private[this] val urls =

clean[UrlEntity](e =>

isNullOrEmpty(e.url) || isNullOrEmpty(e.expanded) || isNullOrEmpty(e.display))

private[this] val media =

Mutation[Seq[MediaEntity]] { mediaEntities =>

mediaEntities.partition(isPartialMedia) match {

case (Nil, nonPartial) => None

case (partial, nonPartial) => Some(nonPartial)

}

}

def isPartialMedia(e: MediaEntity): Boolean =

e.fromIndex < 0 ||

e.toIndex <= 0 ||

isNullOrEmpty(e.url) ||

isNullOrEmpty(e.displayUrl) ||

isNullOrEmpty(e.mediaUrl) ||

isNullOrEmpty(e.mediaUrlHttps) ||

isNullOrEmpty(e.expandedUrl) ||

e.mediaInfo.isEmpty ||

e.mediaKey.isEmpty ||

(MediaKeyClassifier.isImage(MediaKeyUtil.get(e)) && containsInvalidSizeVariant(e.sizes))

private[this] val userMentions =

clean[UserMention](e => e.screenName.isEmpty || e.name.isEmpty)

def isNullOrEmpty(optString: Option[String]): Boolean =

optString.isEmpty || optString.exists(isNullOrEmpty(\_))

def isNullOrEmpty(str: String): Boolean = str == null || str.isEmpty

def containsInvalidSizeVariant(sizes: Set[MediaSize]): Boolean =

sizes.exists(size => size.height == 0 || size.width == 0)

}