package com.twitter.unified\_user\_actions.adapter.tweetypie\_event

import com.twitter.finagle.stats.NullStatsReceiver

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.finatra.kafka.serde.UnKeyed

import com.twitter.unified\_user\_actions.adapter.AbstractAdapter

import com.twitter.tweetypie.thriftscala.TweetEvent

import com.twitter.tweetypie.thriftscala.TweetEventData

import com.twitter.tweetypie.thriftscala.TweetCreateEvent

import com.twitter.tweetypie.thriftscala.TweetDeleteEvent

import com.twitter.tweetypie.thriftscala.TweetEventFlags

import com.twitter.unified\_user\_actions.thriftscala.UnifiedUserAction

class TweetypieEventAdapter extends AbstractAdapter[TweetEvent, UnKeyed, UnifiedUserAction] {

import TweetypieEventAdapter.\_

override def adaptOneToKeyedMany(

tweetEvent: TweetEvent,

statsReceiver: StatsReceiver = NullStatsReceiver

): Seq[(UnKeyed, UnifiedUserAction)] =

adaptEvent(tweetEvent).map(e => (UnKeyed, e))

}

object TweetypieEventAdapter {

def adaptEvent(tweetEvent: TweetEvent): Seq[UnifiedUserAction] = {

Option(tweetEvent).flatMap { e =>

e.data match {

case TweetEventData.TweetCreateEvent(tweetCreateEvent: TweetCreateEvent) =>

getUUAFromTweetCreateEvent(tweetCreateEvent, e.flags)

case TweetEventData.TweetDeleteEvent(tweetDeleteEvent: TweetDeleteEvent) =>

getUUAFromTweetDeleteEvent(tweetDeleteEvent, e.flags)

case \_ => None

}

}.toSeq

}

def getUUAFromTweetCreateEvent(

tweetCreateEvent: TweetCreateEvent,

tweetEventFlags: TweetEventFlags

): Option[UnifiedUserAction] = {

val tweetTypeOpt = TweetypieEventUtils.tweetTypeFromTweet(tweetCreateEvent.tweet)

tweetTypeOpt.flatMap { tweetType =>

tweetType match {

case TweetTypeReply =>

TweetypieReplyEvent.getUnifiedUserAction(tweetCreateEvent, tweetEventFlags)

case TweetTypeRetweet =>

TweetypieRetweetEvent.getUnifiedUserAction(tweetCreateEvent, tweetEventFlags)

case TweetTypeQuote =>

TweetypieQuoteEvent.getUnifiedUserAction(tweetCreateEvent, tweetEventFlags)

case TweetTypeDefault =>

TweetypieCreateEvent.getUnifiedUserAction(tweetCreateEvent, tweetEventFlags)

case TweetTypeEdit =>

TweetypieEditEvent.getUnifiedUserAction(tweetCreateEvent, tweetEventFlags)

}

}

}

def getUUAFromTweetDeleteEvent(

tweetDeleteEvent: TweetDeleteEvent,

tweetEventFlags: TweetEventFlags

): Option[UnifiedUserAction] = {

val tweetTypeOpt = TweetypieEventUtils.tweetTypeFromTweet(tweetDeleteEvent.tweet)

tweetTypeOpt.flatMap { tweetType =>

tweetType match {

case TweetTypeRetweet =>

TweetypieUnretweetEvent.getUnifiedUserAction(tweetDeleteEvent, tweetEventFlags)

case TweetTypeReply =>

TweetypieUnreplyEvent.getUnifiedUserAction(tweetDeleteEvent, tweetEventFlags)

case TweetTypeQuote =>

TweetypieUnquoteEvent.getUnifiedUserAction(tweetDeleteEvent, tweetEventFlags)

case TweetTypeDefault | TweetTypeEdit =>

TweetypieDeleteEvent.getUnifiedUserAction(tweetDeleteEvent, tweetEventFlags)

}

}

}

}