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

import com.twitter.tweetypie.thriftscala.QuotedTweet

import com.twitter.tweetypie.thriftscala.Share

import com.twitter.tweetypie.thriftscala.TweetDeleteEvent

import com.twitter.tweetypie.thriftscala.TweetEventFlags

import com.twitter.unified\_user\_actions.adapter.common.AdapterUtils

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

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

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

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

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

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

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

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

trait BaseTweetypieTweetEventDelete extends BaseTweetypieTweetEvent[TweetDeleteEvent] {

type ExtractedEvent

protected def actionType: ActionType

def getUnifiedUserAction(

tweetDeleteEvent: TweetDeleteEvent,

tweetEventFlags: TweetEventFlags

): Option[UnifiedUserAction] =

extract(tweetDeleteEvent).map { extractedEvent =>

UnifiedUserAction(

userIdentifier = getUserIdentifier(tweetDeleteEvent),

item = getItem(extractedEvent, tweetDeleteEvent),

actionType = actionType,

eventMetadata = getEventMetadata(tweetDeleteEvent, tweetEventFlags)

)

}

protected def extract(tweetDeleteEvent: TweetDeleteEvent): Option[ExtractedEvent]

protected def getItem(extractedEvent: ExtractedEvent, tweetDeleteEvent: TweetDeleteEvent): Item

protected def getUserIdentifier(tweetDeleteEvent: TweetDeleteEvent): UserIdentifier =

UserIdentifier(userId = tweetDeleteEvent.user.map(\_.id))

protected def getEventMetadata(

tweetDeleteEvent: TweetDeleteEvent,

flags: TweetEventFlags

): EventMetadata =

EventMetadata(

sourceTimestampMs = flags.timestampMs,

receivedTimestampMs = AdapterUtils.currentTimestampMs,

sourceLineage = SourceLineage.ServerTweetypieEvents,

traceId = None, // Currently traceId is not stored in TweetDeleteEvent.

// UUA sets this to None since there is no request level language info.

language = None,

// UUA sets this to be consistent with IESource. For the definition,

// see https://sourcegraph.twitter.biz/git.twitter.biz/source/-/blob/src/thrift/com/twitter/tweetypie/tweet.thrift?L1001.

// The definition here conflicts with the intention of UUA to log the request country code

// rather than the signup / geo-tagging country.

countryCode = tweetDeleteEvent.tweet.place.flatMap(\_.countryCode),

/\* clientApplicationId is user's app id if the delete is initiated by a user,

\* or auditor's app id if the delete is initiated by an auditor \*/

clientAppId = tweetDeleteEvent.audit.flatMap(\_.clientApplicationId),

clientVersion = None // Currently clientVersion is not stored in TweetDeleteEvent.

)

}

object TweetypieDeleteEvent extends BaseTweetypieTweetEventDelete {

type ExtractedEvent = Long

override protected val actionType: ActionType = ActionType.ServerTweetDelete

override protected def extract(tweetDeleteEvent: TweetDeleteEvent): Option[Long] = Some(

tweetDeleteEvent.tweet.id)

protected def getItem(

tweetId: Long,

tweetDeleteEvent: TweetDeleteEvent

): Item =

Item.TweetInfo(

TweetInfo(

actionTweetId = tweetId,

actionTweetAuthorInfo =

Some(AuthorInfo(authorId = tweetDeleteEvent.tweet.coreData.map(\_.userId)))

))

}

object TweetypieUnretweetEvent extends BaseTweetypieTweetEventDelete {

override protected val actionType: ActionType = ActionType.ServerTweetUnretweet

override type ExtractedEvent = Share

override protected def extract(tweetDeleteEvent: TweetDeleteEvent): Option[Share] =

tweetDeleteEvent.tweet.coreData.flatMap(\_.share)

override protected def getItem(share: Share, tweetDeleteEvent: TweetDeleteEvent): Item =

Item.TweetInfo(

TweetInfo(

actionTweetId = share.sourceStatusId,

actionTweetAuthorInfo = Some(AuthorInfo(authorId = Some(share.sourceUserId))),

retweetingTweetId = Some(tweetDeleteEvent.tweet.id)

)

)

}

object TweetypieUnreplyEvent extends BaseTweetypieTweetEventDelete {

case class PredicateOutput(tweetId: Long, userId: Long)

override type ExtractedEvent = PredicateOutput

override protected val actionType: ActionType = ActionType.ServerTweetUnreply

override protected def extract(tweetDeleteEvent: TweetDeleteEvent): Option[PredicateOutput] =

tweetDeleteEvent.tweet.coreData

.flatMap(\_.reply).flatMap(r =>

r.inReplyToStatusId.map(tweetId => PredicateOutput(tweetId, r.inReplyToUserId)))

override protected def getItem(

repliedTweet: PredicateOutput,

tweetDeleteEvent: TweetDeleteEvent

): Item = {

Item.TweetInfo(

TweetInfo(

actionTweetId = repliedTweet.tweetId,

actionTweetAuthorInfo = Some(AuthorInfo(authorId = Some(repliedTweet.userId))),

replyingTweetId = Some(tweetDeleteEvent.tweet.id)

)

)

}

}

object TweetypieUnquoteEvent extends BaseTweetypieTweetEventDelete {

override protected val actionType: ActionType = ActionType.ServerTweetUnquote

type ExtractedEvent = QuotedTweet

override protected def extract(tweetDeleteEvent: TweetDeleteEvent): Option[QuotedTweet] =

tweetDeleteEvent.tweet.quotedTweet

override protected def getItem(

quotedTweet: QuotedTweet,

tweetDeleteEvent: TweetDeleteEvent

): Item =

Item.TweetInfo(

TweetInfo(

actionTweetId = quotedTweet.tweetId,

actionTweetAuthorInfo = Some(AuthorInfo(authorId = Some(quotedTweet.userId))),

quotingTweetId = Some(tweetDeleteEvent.tweet.id)

)

)

}