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.TweetCreateEvent

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

/\*\*

\* Base class for Tweetypie TweetCreateEvent including Quote, Reply, Retweet, and Create.

\*/

trait BaseTweetypieTweetEventCreate extends BaseTweetypieTweetEvent[TweetCreateEvent] {

type ExtractedEvent

protected def actionType: ActionType

/\*\*

\* This is the country code where actionTweetId is sent from. For the definitions,

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

\*

\* UUA sets this to be consistent with IESource to meet existing use requirement.

\*

\* For ServerTweetReply/Retweet/Quote, the geo-tagging country code is not available in TweetCreatEvent.

\* Thus, user signup country is picked to meet a customer use case.

\*

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

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

\*

\*/

protected def getCountryCode(tce: TweetCreateEvent): Option[String] = {

tce.tweet.place match {

case Some(p) => p.countryCode

case \_ => tce.user.safety.flatMap(\_.signupCountryCode)

}

}

protected def getItem(

extractedEvent: ExtractedEvent,

tweetCreateEvent: TweetCreateEvent

): Item

protected def extract(tweetCreateEvent: TweetCreateEvent): Option[ExtractedEvent]

def getUnifiedUserAction(

tweetCreateEvent: TweetCreateEvent,

tweetEventFlags: TweetEventFlags

): Option[UnifiedUserAction] = {

extract(tweetCreateEvent).map { extractedEvent =>

UnifiedUserAction(

userIdentifier = getUserIdentifier(tweetCreateEvent),

item = getItem(extractedEvent, tweetCreateEvent),

actionType = actionType,

eventMetadata = getEventMetadata(tweetCreateEvent, tweetEventFlags),

productSurface = None,

productSurfaceInfo = None

)

}

}

protected def getUserIdentifier(tweetCreateEvent: TweetCreateEvent): UserIdentifier =

UserIdentifier(userId = Some(tweetCreateEvent.user.id))

protected def getEventMetadata(

tweetCreateEvent: TweetCreateEvent,

flags: TweetEventFlags

): EventMetadata =

EventMetadata(

sourceTimestampMs = flags.timestampMs,

receivedTimestampMs = AdapterUtils.currentTimestampMs,

sourceLineage = SourceLineage.ServerTweetypieEvents,

traceId = None, // Currently traceId is not stored in TweetCreateEvent

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

language = None,

countryCode = getCountryCode(tweetCreateEvent),

clientAppId = tweetCreateEvent.tweet.deviceSource.flatMap(\_.clientAppId),

clientVersion = None // Currently clientVersion is not stored in TweetCreateEvent

)

}

/\*\*

\* Get UnifiedUserAction from a tweet Create.

\* Note the Create is generated when the tweet is not a Quote/Retweet/Reply.

\*/

object TweetypieCreateEvent extends BaseTweetypieTweetEventCreate {

type ExtractedEvent = Long

override protected val actionType: ActionType = ActionType.ServerTweetCreate

override protected def extract(tweetCreateEvent: TweetCreateEvent): Option[Long] =

Option(tweetCreateEvent.tweet.id)

protected def getItem(

tweetId: Long,

tweetCreateEvent: TweetCreateEvent

): Item =

Item.TweetInfo(

TweetInfo(

actionTweetId = tweetId,

actionTweetAuthorInfo = Some(AuthorInfo(authorId = Some(tweetCreateEvent.user.id)))

))

}

/\*\*

\* Get UnifiedUserAction from a Reply.

\* Note the Reply is generated when someone is replying to a tweet.

\*/

object TweetypieReplyEvent extends BaseTweetypieTweetEventCreate {

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

override type ExtractedEvent = PredicateOutput

override protected val actionType: ActionType = ActionType.ServerTweetReply

override protected def extract(tweetCreateEvent: TweetCreateEvent): Option[PredicateOutput] =

tweetCreateEvent.tweet.coreData

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

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

override protected def getItem(

repliedTweet: PredicateOutput,

tweetCreateEvent: TweetCreateEvent

): Item = {

Item.TweetInfo(

TweetInfo(

actionTweetId = repliedTweet.tweetId,

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

replyingTweetId = Some(tweetCreateEvent.tweet.id)

)

)

}

}

/\*\*

\* Get UnifiedUserAction from a Quote.

\* Note the Quote is generated when someone is quoting (retweeting with comment) a tweet.

\*/

object TweetypieQuoteEvent extends BaseTweetypieTweetEventCreate {

override protected val actionType: ActionType = ActionType.ServerTweetQuote

type ExtractedEvent = QuotedTweet

override protected def extract(tweetCreateEvent: TweetCreateEvent): Option[QuotedTweet] =

tweetCreateEvent.tweet.quotedTweet

override protected def getItem(

quotedTweet: QuotedTweet,

tweetCreateEvent: TweetCreateEvent

): Item =

Item.TweetInfo(

TweetInfo(

actionTweetId = quotedTweet.tweetId,

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

quotingTweetId = Some(tweetCreateEvent.tweet.id)

)

)

}

/\*\*

\* Get UnifiedUserAction from a Retweet.

\* Note the Retweet is generated when someone is retweeting (without comment) a tweet.

\*/

object TweetypieRetweetEvent extends BaseTweetypieTweetEventCreate {

override type ExtractedEvent = Share

override protected val actionType: ActionType = ActionType.ServerTweetRetweet

override protected def extract(tweetCreateEvent: TweetCreateEvent): Option[Share] =

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

override protected def getItem(share: Share, tweetCreateEvent: TweetCreateEvent): Item =

Item.TweetInfo(

TweetInfo(

actionTweetId = share.sourceStatusId,

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

retweetingTweetId = Some(tweetCreateEvent.tweet.id)

)

)

}

/\*\*

\* Get UnifiedUserAction from a TweetEdit.

\* Note the Edit is generated when someone is editing their quote or default tweet. The edit will

\* generate a new Tweet.

\*/

object TweetypieEditEvent extends BaseTweetypieTweetEventCreate {

override type ExtractedEvent = Long

override protected def actionType: ActionType = ActionType.ServerTweetEdit

override protected def extract(tweetCreateEvent: TweetCreateEvent): Option[Long] =

TweetypieEventUtils.editedTweetIdFromTweet(tweetCreateEvent.tweet)

override protected def getItem(

editedTweetId: Long,

tweetCreateEvent: TweetCreateEvent

): Item =

Item.TweetInfo(

TweetInfo(

actionTweetId = tweetCreateEvent.tweet.id,

actionTweetAuthorInfo = Some(AuthorInfo(authorId = Some(tweetCreateEvent.user.id))),

editedTweetId = Some(editedTweetId),

quotedTweetId = tweetCreateEvent.tweet.quotedTweet.map(\_.tweetId)

)

)

}