package com.twitter.unified\_user\_actions.adapter.retweet\_archival\_events

import com.twitter.finagle.stats.NullStatsReceiver

import com.twitter.finagle.stats.StatsReceiver

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

import com.twitter.tweetypie.thriftscala.RetweetArchivalEvent

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

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

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

class RetweetArchivalEventsAdapter

extends AbstractAdapter[RetweetArchivalEvent, UnKeyed, UnifiedUserAction] {

import RetweetArchivalEventsAdapter.\_

override def adaptOneToKeyedMany(

input: RetweetArchivalEvent,

statsReceiver: StatsReceiver = NullStatsReceiver

): Seq[(UnKeyed, UnifiedUserAction)] =

adaptEvent(input).map { e => (UnKeyed, e) }

}

object RetweetArchivalEventsAdapter {

def adaptEvent(e: RetweetArchivalEvent): Seq[UnifiedUserAction] =

Option(e).map { e =>

UnifiedUserAction(

userIdentifier = UserIdentifier(userId = Some(e.retweetUserId)),

item = getItem(e),

actionType =

if (e.isArchivingAction.getOrElse(true)) ActionType.ServerTweetArchiveRetweet

else ActionType.ServerTweetUnarchiveRetweet,

eventMetadata = getEventMetadata(e)

)

}.toSeq

def getItem(e: RetweetArchivalEvent): Item =

Item.TweetInfo(

TweetInfo(

actionTweetId = e.srcTweetId,

actionTweetAuthorInfo = Some(AuthorInfo(authorId = Some(e.srcTweetUserId))),

retweetingTweetId = Some(e.retweetId)

)

)

def getEventMetadata(e: RetweetArchivalEvent): EventMetadata =

EventMetadata(

sourceTimestampMs = e.timestampMs,

receivedTimestampMs = AdapterUtils.currentTimestampMs,

sourceLineage = SourceLineage.ServerRetweetArchivalEvents,

)

}