package com.twitter.unified\_user\_actions.adapter.uua\_aggregates

import com.twitter.finagle.stats.NullStatsReceiver

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.iesource.thriftscala.ClientEventContext

import com.twitter.iesource.thriftscala.EngagingContext

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

import com.twitter.iesource.thriftscala.InteractionType

import com.twitter.iesource.thriftscala.InteractionEvent

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

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

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

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

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

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

/\*\*

\* This is to read directly from InteractionEvents

\*/

class RekeyUuaFromInteractionEventsAdapter

extends AbstractAdapter[InteractionEvent, Long, KeyedUuaTweet] {

import RekeyUuaFromInteractionEventsAdapter.\_

override def adaptOneToKeyedMany(

input: InteractionEvent,

statsReceiver: StatsReceiver = NullStatsReceiver

): Seq[(Long, KeyedUuaTweet)] =

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

}

object RekeyUuaFromInteractionEventsAdapter {

def adaptEvent(

e: InteractionEvent,

statsReceiver: StatsReceiver = NullStatsReceiver

): Seq[KeyedUuaTweet] =

Option(e).flatMap { e =>

e.interactionType.flatMap {

case InteractionType.TweetRenderImpression if !isDetailImpression(e.engagingContext) =>

getRekeyedUUA(

input = e,

actionType = ActionType.ClientTweetRenderImpression,

sourceLineage = SourceLineage.ClientEvents,

statsReceiver = statsReceiver)

case \_ => None

}

}.toSeq

def getRekeyedUUA(

input: InteractionEvent,

actionType: ActionType,

sourceLineage: SourceLineage,

statsReceiver: StatsReceiver = NullStatsReceiver

): Option[KeyedUuaTweet] =

input.engagingUserId match {

// please see https://docs.google.com/document/d/1-fy2S-8-YMRQgEN0Sco0OLTmeOIUdqgiZ5G1KwTHt2g/edit#

// in order to withstand of potential attacks, we filter out the logged-out users.

// Checking user id is 0 is the reverse engineering of

// https://sourcegraph.twitter.biz/git.twitter.biz/source/-/blob/iesource/thrift/src/main/thrift/com/twitter/iesource/interaction\_event.thrift?L220

// https://sourcegraph.twitter.biz/git.twitter.biz/source/-/blob/iesource/common/src/main/scala/com/twitter/iesource/common/converters/client/LogEventConverter.scala?L198

case 0L =>

statsReceiver.counter("loggedOutEvents").incr()

None

case \_ =>

Some(

KeyedUuaTweet(

tweetId = input.targetId,

actionType = actionType,

userIdentifier = UserIdentifier(userId = Some(input.engagingUserId)),

eventMetadata = EventMetadata(

sourceTimestampMs = input.triggeredTimestampMillis.getOrElse(input.timestampMillis),

receivedTimestampMs = AdapterUtils.currentTimestampMs,

sourceLineage = sourceLineage

)

))

}

def isDetailImpression(engagingContext: EngagingContext): Boolean =

engagingContext match {

case EngagingContext.ClientEventContext(

ClientEventContext(\_, \_, \_, \_, \_, \_, \_, Some(isDetailsImpression), \_)

) if isDetailsImpression =>

true

case \_ => false

}

}