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

import com.twitter.clientapp.thriftscala.ItemType

import com.twitter.clientapp.thriftscala.LogEvent

import com.twitter.clientapp.thriftscala.{Item => LogEventItem}

import com.twitter.logbase.thriftscala.ClientEventReceiver

import com.twitter.logbase.thriftscala.LogBase

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

abstract class BaseClientEvent(actionType: ActionType) {

def toUnifiedUserAction(logEvent: LogEvent): Seq[UnifiedUserAction] = {

val logBase: Option[LogBase] = logEvent.logBase

for {

ed <- logEvent.eventDetails.toSeq

items <- ed.items.toSeq

ceItem <- items

eventTimestamp <- logBase.flatMap(getSourceTimestamp)

uuaItem <- getUuaItem(ceItem, logEvent)

if isItemTypeValid(ceItem.itemType)

} yield {

val userIdentifier: UserIdentifier = UserIdentifier(

userId = logBase.flatMap(\_.userId),

guestIdMarketing = logBase.flatMap(\_.guestIdMarketing))

val productSurface: Option[ProductSurface] = ProductSurfaceUtils

.getProductSurface(logEvent.eventNamespace)

val eventMetaData: EventMetadata = ClientEventCommonUtils

.getEventMetadata(

eventTimestamp = eventTimestamp,

logEvent = logEvent,

ceItem = ceItem,

productSurface = productSurface

)

UnifiedUserAction(

userIdentifier = userIdentifier,

item = uuaItem,

actionType = actionType,

eventMetadata = eventMetaData,

productSurface = productSurface,

productSurfaceInfo =

ProductSurfaceUtils.getProductSurfaceInfo(productSurface, ceItem, logEvent)

)

}

}

def getUuaItem(

ceItem: LogEventItem,

logEvent: LogEvent

): Option[Item] = for (actionTweetId <- ceItem.id)

yield Item.TweetInfo(

ClientEventCommonUtils

.getBasicTweetInfo(actionTweetId, ceItem, logEvent.eventNamespace))

// default implementation filters items of type tweet

// override in the subclass implementation to filter items of other types

def isItemTypeValid(itemTypeOpt: Option[ItemType]): Boolean =

ItemTypeFilterPredicates.isItemTypeTweet(itemTypeOpt)

def getSourceTimestamp(logBase: LogBase): Option[Long] =

logBase.clientEventReceiver match {

case Some(ClientEventReceiver.CesHttp) | Some(ClientEventReceiver.CesThrift) =>

logBase.driftAdjustedEventCreatedAtMs

case \_ => Some(logBase.driftAdjustedEventCreatedAtMs.getOrElse(logBase.timestamp))

}

}