package com.twitter.unified\_user\_actions.adapter.tls\_favs\_event

import com.twitter.finagle.stats.NullStatsReceiver

import com.twitter.finagle.stats.StatsReceiver

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

import com.twitter.timelineservice.thriftscala.\_

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 TlsFavsAdapter

extends AbstractAdapter[ContextualizedFavoriteEvent, UnKeyed, UnifiedUserAction] {

import TlsFavsAdapter.\_

override def adaptOneToKeyedMany(

input: ContextualizedFavoriteEvent,

statsReceiver: StatsReceiver = NullStatsReceiver

): Seq[(UnKeyed, UnifiedUserAction)] =

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

}

object TlsFavsAdapter {

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

Option(e).flatMap { e =>

e.event match {

case FavoriteEventUnion.Favorite(favoriteEvent) =>

Some(

UnifiedUserAction(

userIdentifier = getUserIdentifier(Left(favoriteEvent)),

item = getFavItem(favoriteEvent),

actionType = ActionType.ServerTweetFav,

eventMetadata = getEventMetadata(Left(favoriteEvent), e.context),

productSurface = None,

productSurfaceInfo = None

))

case FavoriteEventUnion.Unfavorite(unfavoriteEvent) =>

Some(

UnifiedUserAction(

userIdentifier = getUserIdentifier(Right(unfavoriteEvent)),

item = getUnfavItem(unfavoriteEvent),

actionType = ActionType.ServerTweetUnfav,

eventMetadata = getEventMetadata(Right(unfavoriteEvent), e.context),

productSurface = None,

productSurfaceInfo = None

))

case \_ => None

}

}.toSeq

def getFavItem(favoriteEvent: FavoriteEvent): Item =

Item.TweetInfo(

TweetInfo(

actionTweetId = favoriteEvent.tweetId,

actionTweetAuthorInfo = Some(AuthorInfo(authorId = Some(favoriteEvent.tweetUserId))),

retweetingTweetId = favoriteEvent.retweetId

)

)

def getUnfavItem(unfavoriteEvent: UnfavoriteEvent): Item =

Item.TweetInfo(

TweetInfo(

actionTweetId = unfavoriteEvent.tweetId,

actionTweetAuthorInfo = Some(AuthorInfo(authorId = Some(unfavoriteEvent.tweetUserId))),

retweetingTweetId = unfavoriteEvent.retweetId

)

)

def getEventMetadata(

event: Either[FavoriteEvent, UnfavoriteEvent],

context: LogEventContext

): EventMetadata = {

val sourceTimestampMs = event match {

case Left(favoriteEvent) => favoriteEvent.eventTimeMs

case Right(unfavoriteEvent) => unfavoriteEvent.eventTimeMs

}

// Client UI language, see more at http://go/languagepriority. The format should be ISO 639-1.

val language = event match {

case Left(favoriteEvent) => favoriteEvent.viewerContext.flatMap(\_.requestLanguageCode)

case Right(unfavoriteEvent) => unfavoriteEvent.viewerContext.flatMap(\_.requestLanguageCode)

}

// From the request (user’s current location),

// see https://sourcegraph.twitter.biz/git.twitter.biz/source/-/blob/src/thrift/com/twitter/context/viewer.thrift?L54

// The format should be ISO\_3166-1\_alpha-2.

val countryCode = event match {

case Left(favoriteEvent) => favoriteEvent.viewerContext.flatMap(\_.requestCountryCode)

case Right(unfavoriteEvent) => unfavoriteEvent.viewerContext.flatMap(\_.requestCountryCode)

}

EventMetadata(

sourceTimestampMs = sourceTimestampMs,

receivedTimestampMs = AdapterUtils.currentTimestampMs,

sourceLineage = SourceLineage.ServerTlsFavs,

language = language.map(AdapterUtils.normalizeLanguageCode),

countryCode = countryCode.map(AdapterUtils.normalizeCountryCode),

traceId = Some(context.traceId),

clientAppId = context.clientApplicationId,

)

}

// Get id of the user that took the action

def getUserIdentifier(event: Either[FavoriteEvent, UnfavoriteEvent]): UserIdentifier =

event match {

case Left(favoriteEvent) => UserIdentifier(userId = Some(favoriteEvent.userId))

case Right(unfavoriteEvent) => UserIdentifier(userId = Some(unfavoriteEvent.userId))

}

}