package com.twitter.product\_mixer.component\_library.decorator.urt.builder.item.ad

import com.twitter.ads.adserver.{thriftscala => ads}

import com.twitter.adserver.{thriftscala => adserver}

import com.twitter.product\_mixer.component\_library.decorator.urt.builder.contextual\_ref.ContextualTweetRefBuilder

import com.twitter.product\_mixer.component\_library.decorator.urt.builder.item.tweet.TweetCandidateUrtItemBuilder.TweetClientEventInfoElement

import com.twitter.product\_mixer.component\_library.model.candidate.ads.AdsCandidate

import com.twitter.product\_mixer.component\_library.model.candidate.ads.AdsTweetCandidate

import com.twitter.product\_mixer.core.feature.featuremap.FeatureMap

import com.twitter.product\_mixer.core.functional\_component.decorator.urt.builder.CandidateUrtEntryBuilder

import com.twitter.product\_mixer.core.functional\_component.decorator.urt.builder.metadata.BaseClientEventInfoBuilder

import com.twitter.product\_mixer.core.model.marshalling.response.urt.TimelineItem

import com.twitter.product\_mixer.core.model.marshalling.response.urt.item.tweet.Tweet

import com.twitter.product\_mixer.core.model.marshalling.response.urt.item.tweet.TweetDisplayType

import com.twitter.product\_mixer.core.model.marshalling.response.urt.item.tweet.TweetItem

import com.twitter.product\_mixer.core.model.marshalling.response.urt.promoted.AdMetadataContainer

import com.twitter.product\_mixer.core.model.marshalling.response.urt.promoted.Amplify

import com.twitter.product\_mixer.core.model.marshalling.response.urt.promoted.CallToAction

import com.twitter.product\_mixer.core.model.marshalling.response.urt.promoted.ClickTrackingInfo

import com.twitter.product\_mixer.core.model.marshalling.response.urt.promoted.DcmUrlOverrideType

import com.twitter.product\_mixer.core.model.marshalling.response.urt.promoted.DirectSponsorshipType

import com.twitter.product\_mixer.core.model.marshalling.response.urt.promoted.DisclaimerIssue

import com.twitter.product\_mixer.core.model.marshalling.response.urt.promoted.DisclaimerPolitical

import com.twitter.product\_mixer.core.model.marshalling.response.urt.promoted.DisclaimerType

import com.twitter.product\_mixer.core.model.marshalling.response.urt.promoted.DisclosureType

import com.twitter.product\_mixer.core.model.marshalling.response.urt.promoted.DynamicPrerollType

import com.twitter.product\_mixer.core.model.marshalling.response.urt.promoted.Earned

import com.twitter.product\_mixer.core.model.marshalling.response.urt.promoted.IndirectSponsorshipType

import com.twitter.product\_mixer.core.model.marshalling.response.urt.promoted.Issue

import com.twitter.product\_mixer.core.model.marshalling.response.urt.promoted.LiveTvEvent

import com.twitter.product\_mixer.core.model.marshalling.response.urt.promoted.Marketplace

import com.twitter.product\_mixer.core.model.marshalling.response.urt.promoted.MediaInfo

import com.twitter.product\_mixer.core.model.marshalling.response.urt.promoted.NoDisclosure

import com.twitter.product\_mixer.core.model.marshalling.response.urt.promoted.NoSponsorshipSponsorshipType

import com.twitter.product\_mixer.core.model.marshalling.response.urt.promoted.Political

import com.twitter.product\_mixer.core.model.marshalling.response.urt.promoted.Preroll

import com.twitter.product\_mixer.core.model.marshalling.response.urt.promoted.PrerollMetadata

import com.twitter.product\_mixer.core.model.marshalling.response.urt.promoted.PromotedMetadata

import com.twitter.product\_mixer.core.model.marshalling.response.urt.promoted.SkAdNetworkData

import com.twitter.product\_mixer.core.model.marshalling.response.urt.promoted.SponsorshipType

import com.twitter.product\_mixer.core.model.marshalling.response.urt.promoted.UnknownUrlOverrideType

import com.twitter.product\_mixer.core.model.marshalling.response.urt.promoted.VideoVariant

import com.twitter.product\_mixer.core.pipeline.PipelineQuery

import com.twitter.timelines.util.AdMetadataContainerSerializer

import com.twitter.timelines.util.PrerollMetadataSerializer

/\*\*

\* [[AdsCandidateUrtItemBuilder]] takes a [[AdsCandidate]] (with a [[Query]] as additional context)

\* and converts it into the Product Mixer URT representation, or throws an error.

\*

\* Currently, the only supported form for URT representation of the [[AdsCandidate]] is a [[Tweet]],

\* but in the future it could be expanded to handle other forms of ads.

\*

\* @param tweetClientEventInfoBuilder Optionally, provide a ClientEventInfoBuilder for Tweets

\* that given an AdsTweetCandidate and element of "tweet".

\* @param tweetDisplayType Should be [[EmphasizedPromotedTweet]] on Profile timelines,

\* otherwise [[Tweet]]

\*/

case class AdsCandidateUrtItemBuilder[Query <: PipelineQuery](

tweetClientEventInfoBuilder: Option[BaseClientEventInfoBuilder[Query, AdsTweetCandidate]] = None,

contextualTweetRefBuilder: Option[ContextualTweetRefBuilder[AdsTweetCandidate]] = None,

tweetDisplayType: TweetDisplayType = Tweet)

extends CandidateUrtEntryBuilder[Query, AdsCandidate, TimelineItem] {

override def apply(

pipelineQuery: Query,

candidate: AdsCandidate,

candidateFeatures: FeatureMap

): TimelineItem = {

candidate match {

case tweetCandidate: AdsTweetCandidate =>

TweetItem(

id = tweetCandidate.id,

entryNamespace = TweetItem.PromotedTweetEntryNamespace,

sortIndex = None, // Sort indexes are automatically set in the domain marshaller phase

clientEventInfo = tweetClientEventInfoBuilder.flatMap(

\_.apply(

pipelineQuery,

tweetCandidate,

candidateFeatures,

Some(TweetClientEventInfoElement))),

feedbackActionInfo = None,

isPinned = None,

entryIdToReplace = None,

socialContext = None,

highlights = None,

innerTombstoneInfo = None,

timelinesScoreInfo = None,

hasModeratedReplies = None,

forwardPivot = None,

innerForwardPivot = None,

conversationAnnotation = None,

promotedMetadata = Some(promotedMetadata(tweetCandidate.adImpression)),

displayType = tweetDisplayType,

contextualTweetRef = contextualTweetRefBuilder.flatMap(\_.apply(tweetCandidate)),

prerollMetadata = prerollMetadata(tweetCandidate.adImpression),

replyBadge = None,

destination = None

)

}

}

private def promotedMetadata(impression: adserver.AdImpression) = {

PromotedMetadata(

advertiserId = impression.advertiserId,

impressionString = impression.impressionString,

disclosureType = impression.disclosureType.map(convertDisclosureType),

experimentValues = impression.experimentValues.map(\_.toMap),

promotedTrendId = impression.promotedTrendId.map(\_.toLong),

promotedTrendName = impression.promotedTrendName,

promotedTrendQueryTerm = impression.promotedTrendQueryTerm,

promotedTrendDescription = impression.promotedTrendDescription,

clickTrackingInfo = impression.clickTrackingInfo.map(convertClickTrackingInfo),

adMetadataContainer = adMetadataContainer(impression)

)

}

private def convertDisclosureType(

disclosureType: adserver.DisclosureType

): DisclosureType = disclosureType match {

case adserver.DisclosureType.None => NoDisclosure

case adserver.DisclosureType.Political => Political

case adserver.DisclosureType.Earned => Earned

case adserver.DisclosureType.Issue => Issue

case \_ => throw new UnsupportedDisclosureTypeException(disclosureType)

}

private def convertClickTrackingInfo(

clickTracking: adserver.ClickTrackingInfo

): ClickTrackingInfo = ClickTrackingInfo(

urlParams = clickTracking.urlParams.getOrElse(Map.empty),

urlOverride = clickTracking.urlOverride,

urlOverrideType = clickTracking.urlOverrideType.map {

case adserver.UrlOverrideType.Unknown => UnknownUrlOverrideType

case adserver.UrlOverrideType.Dcm => DcmUrlOverrideType

case \_ => throw new UnsupportedClickTrackingInfoException(clickTracking)

}

)

private def prerollMetadata(adImpression: adserver.AdImpression): Option[PrerollMetadata] = {

adImpression.serializedPrerollMetadata

.flatMap(PrerollMetadataSerializer.deserialize).map { metadata =>

PrerollMetadata(

metadata.preroll.map(convertPreroll),

metadata.videoAnalyticsScribePassthrough

)

}

}

private def adMetadataContainer(

adImpression: adserver.AdImpression

): Option[AdMetadataContainer] = {

adImpression.serializedAdMetadataContainer

.flatMap(AdMetadataContainerSerializer.deserialize).map { container =>

AdMetadataContainer(

removePromotedAttributionForPreroll = container.removePromotedAttributionForPreroll,

sponsorshipCandidate = container.sponsorshipCandidate,

sponsorshipOrganization = container.sponsorshipOrganization,

sponsorshipOrganizationWebsite = container.sponsorshipOrganizationWebsite,

sponsorshipType = container.sponsorshipType.map(convertSponsorshipType),

disclaimerType = container.disclaimerType.map(convertDisclaimerType),

skAdNetworkDataList = container.skAdNetworkDataList.map(convertSkAdNetworkDataList),

unifiedCardOverride = container.unifiedCardOverride

)

}

}

private def convertSponsorshipType(

sponsorshipType: ads.SponsorshipType

): SponsorshipType = sponsorshipType match {

case ads.SponsorshipType.Direct => DirectSponsorshipType

case ads.SponsorshipType.Indirect => IndirectSponsorshipType

case ads.SponsorshipType.NoSponsorship => NoSponsorshipSponsorshipType

// Thrift has extras (e.g. Sponsorship4) that are not used in practice

case \_ => throw new UnsupportedSponsorshipTypeException(sponsorshipType)

}

private def convertDisclaimerType(

disclaimerType: ads.DisclaimerType

): DisclaimerType = disclaimerType match {

case ads.DisclaimerType.Political => DisclaimerPolitical

case ads.DisclaimerType.Issue => DisclaimerIssue

case \_ => throw new UnsupportedDisclaimerTypeException(disclaimerType)

}

private def convertDynamicPrerollType(

dynamicPrerollType: ads.DynamicPrerollType

): DynamicPrerollType =

dynamicPrerollType match {

case ads.DynamicPrerollType.Amplify => Amplify

case ads.DynamicPrerollType.Marketplace => Marketplace

case ads.DynamicPrerollType.LiveTvEvent => LiveTvEvent

case \_ => throw new UnsupportedDynamicPrerollTypeException(dynamicPrerollType)

}

private def convertMediaInfo(mediaInfo: ads.MediaInfo): MediaInfo = {

MediaInfo(

uuid = mediaInfo.uuid,

publisherId = mediaInfo.publisherId,

callToAction = mediaInfo.callToAction.map(convertCallToAction),

durationMillis = mediaInfo.durationMillis,

videoVariants = mediaInfo.videoVariants.map(convertVideoVariants),

advertiserName = mediaInfo.advertiserName,

renderAdByAdvertiserName = mediaInfo.renderAdByAdvertiserName,

advertiserProfileImageUrl = mediaInfo.advertiserProfileImageUrl

)

}

private def convertVideoVariants(videoVariants: Seq[ads.VideoVariant]): Seq[VideoVariant] = {

videoVariants.map(videoVariant =>

VideoVariant(

url = videoVariant.url,

contentType = videoVariant.contentType,

bitrate = videoVariant.bitrate

))

}

private def convertCallToAction(callToAction: ads.CallToAction): CallToAction = {

CallToAction(

callToActionType = callToAction.callToActionType,

url = callToAction.url

)

}

private def convertPreroll(

preroll: ads.Preroll

): Preroll = {

Preroll(

preroll.prerollId,

preroll.dynamicPrerollType.map(convertDynamicPrerollType),

preroll.mediaInfo.map(convertMediaInfo)

)

}

private def convertSkAdNetworkDataList(

skAdNetworkDataList: Seq[ads.SkAdNetworkData]

): Seq[SkAdNetworkData] = skAdNetworkDataList.map(sdAdNetwork =>

SkAdNetworkData(

version = sdAdNetwork.version,

srcAppId = sdAdNetwork.srcAppId,

dstAppId = sdAdNetwork.dstAppId,

adNetworkId = sdAdNetwork.adNetworkId,

campaignId = sdAdNetwork.campaignId,

impressionTimeInMillis = sdAdNetwork.impressionTimeInMillis,

nonce = sdAdNetwork.nonce,

signature = sdAdNetwork.signature,

fidelityType = sdAdNetwork.fidelityType

))

}

class UnsupportedClickTrackingInfoException(clickTrackingInfo: adserver.ClickTrackingInfo)

extends UnsupportedOperationException(

s"Unsupported ClickTrackingInfo: $clickTrackingInfo"

)

class UnsupportedDisclaimerTypeException(disclaimerType: ads.DisclaimerType)

extends UnsupportedOperationException(

s"Unsupported DisclaimerType: $disclaimerType"

)

class UnsupportedDisclosureTypeException(disclosureType: adserver.DisclosureType)

extends UnsupportedOperationException(

s"Unsupported DisclosureType: $disclosureType"

)

class UnsupportedDynamicPrerollTypeException(dynamicPrerollType: ads.DynamicPrerollType)

extends UnsupportedOperationException(

s"Unsupported DynamicPrerollType: $dynamicPrerollType"

)

class UnsupportedSponsorshipTypeException(sponsorshipType: ads.SponsorshipType)

extends UnsupportedOperationException(

s"Unsupported SponsorshipType: $sponsorshipType"

)