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

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

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

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

import com.twitter.product\_mixer.core.model.marshalling.response.urt.contextual\_ref.ContextualTweetRef

import com.twitter.product\_mixer.core.model.marshalling.response.urt.item.conversation\_annotation.ConversationAnnotation

import com.twitter.product\_mixer.core.model.marshalling.response.urt.item.forward\_pivot.ForwardPivot

import com.twitter.product\_mixer.core.model.marshalling.response.urt.item.tombstone.TombstoneInfo

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

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

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

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

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.metadata.Url

object TweetItem {

val TweetEntryNamespace = EntryNamespace("tweet")

val PromotedTweetEntryNamespace = EntryNamespace("promoted-tweet")

}

case class TweetItem(

override val id: Long,

override val entryNamespace: EntryNamespace,

override val sortIndex: Option[Long],

override val clientEventInfo: Option[ClientEventInfo],

override val feedbackActionInfo: Option[FeedbackActionInfo],

override val isPinned: Option[Boolean],

override val entryIdToReplace: Option[String],

socialContext: Option[SocialContext],

highlights: Option[TweetHighlights],

displayType: TweetDisplayType,

innerTombstoneInfo: Option[TombstoneInfo],

timelinesScoreInfo: Option[TimelinesScoreInfo],

hasModeratedReplies: Option[Boolean],

forwardPivot: Option[ForwardPivot],

innerForwardPivot: Option[ForwardPivot],

promotedMetadata: Option[PromotedMetadata],

conversationAnnotation: Option[ConversationAnnotation],

contextualTweetRef: Option[ContextualTweetRef],

prerollMetadata: Option[PrerollMetadata],

replyBadge: Option[Badge],

destination: Option[Url])

extends TimelineItem {

/\*\*

\* Promoted tweets need to include the impression ID in the entry ID since some clients have

\* client-side logic that deduplicates ads impression callbacks based on a combination of the

\* tweet and impression IDs. Not including the impression ID will lead to over deduplication.

\*/

override lazy val entryIdentifier: String = promotedMetadata

.map { metadata =>

val impressionId = metadata.impressionString match {

case Some(impressionString) if impressionString.nonEmpty => impressionString

case \_ => throw new IllegalStateException(s"Promoted Tweet $id missing impression ID")

}

s"$entryNamespace-$id-$impressionId"

}.getOrElse(s"$entryNamespace-$id")

override def withSortIndex(sortIndex: Long): TimelineEntry = copy(sortIndex = Some(sortIndex))

}