package com.twitter.home\_mixer.model

import com.twitter.home\_mixer.functional\_component.candidate\_source.EarlybirdBottomTweetFeature

import com.twitter.home\_mixer.functional\_component.candidate\_source.EarlybirdResponseTruncatedFeature

import com.twitter.product\_mixer.component\_library.model.cursor.UrtOrderedCursor

import com.twitter.product\_mixer.component\_library.premarshaller.urt.builder.IncludeInstruction

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

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

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

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

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

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

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

/\*\*

\* Determine whether to include a Gap Cursor in the response based on whether a timeline

\* is truncated because it has more entries than the max response size.

\* There are two ways this can happen:

\* 1) There are unused entries in Earlybird. This is determined by a flag returned from Earlybird.

\* We respect the Earlybird flag only if there are some entries after deduping and filtering

\* to ensure that we do not get stuck repeatedly serving gaps which lead to no tweets.

\* 2) Ads injection can take the response size over the max count. Goldfinch truncates tweet

\* entries in this case. We can check if the bottom tweet from Earlybird is in the response to

\* determine if all Earlybird tweets have been used.

\*

\* While scrolling down to get older tweets (BottomCursor), responses will generally be

\* truncated, but we don't want to render a gap cursor there, so we need to ensure we only

\* apply the truncation check to newer (TopCursor) or middle (GapCursor) requests.

\*

\* We return either a Gap Cursor or a Bottom Cursor, but not both, so the include instruction

\* for Bottom should be the inverse of Gap.

\*/

object GapIncludeInstruction

extends IncludeInstruction[PipelineQuery with HasPipelineCursor[UrtOrderedCursor]] {

override def apply(

query: PipelineQuery with HasPipelineCursor[UrtOrderedCursor],

entries: Seq[TimelineEntry]

): Boolean = {

val wasTruncated = query.features.exists(\_.getOrElse(EarlybirdResponseTruncatedFeature, false))

// Get oldest tweet or tweets within oldest conversation module

val tweetEntries = entries.view.reverse

.collectFirst {

case item: TweetItem if item.promotedMetadata.isEmpty => Seq(item.id.toString)

case module: TimelineModule if module.items.head.item.isInstanceOf[TweetItem] =>

module.items.map(\_.item.id.toString)

}.toSeq.flatten

val bottomCursor =

query.features.flatMap(\_.getOrElse(EarlybirdBottomTweetFeature, None)).map(\_.toString)

// Ads truncation happened if we have at least max count entries and bottom tweet is missing

val adsTruncation = query.requestedMaxResults.exists(\_ <= entries.size) &&

!bottomCursor.exists(tweetEntries.contains)

query.pipelineCursor.exists(\_.cursorType match {

case Some(TopCursor) | Some(GapCursor) =>

(wasTruncated && tweetEntries.nonEmpty) || adsTruncation

case \_ => false

})

}

}