package com.twitter.product\_mixer.component\_library.premarshaller.urt.builder

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

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

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

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

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

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

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

import com.twitter.timelines.configapi.Param

trait BaseUnorderedExcludeIdsBottomCursorBuilder

extends UrtCursorBuilder[

PipelineQuery with HasPipelineCursor[UrtUnorderedExcludeIdsCursor]

] {

def excludedIdsMaxLengthParam: Param[Int]

def excludeEntriesCollector(entries: Seq[TimelineEntry]): Seq[Long]

def serializer: PipelineCursorSerializer[UrtUnorderedExcludeIdsCursor]

override val cursorType: CursorType = BottomCursor

override def cursorValue(

query: PipelineQuery with HasPipelineCursor[UrtUnorderedExcludeIdsCursor],

entries: Seq[TimelineEntry]

): String = {

val excludedIdsMaxLength = query.params(excludedIdsMaxLengthParam)

assert(excludedIdsMaxLength > 0, "Excluded IDs max length must be greater than zero")

val newEntryIds = excludeEntriesCollector(entries)

assert(

newEntryIds.length < excludedIdsMaxLength,

"New entry IDs length must be smaller than excluded IDs max length")

val excludedIds = query.pipelineCursor

.map(\_.excludedIds ++ newEntryIds)

.getOrElse(newEntryIds)

.takeRight(excludedIdsMaxLength)

val cursor = UrtUnorderedExcludeIdsCursor(

initialSortIndex = nextBottomInitialSortIndex(query, entries),

excludedIds = excludedIds

)

serializer.serializeCursor(cursor)

}

}