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

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

import com.twitter.product\_mixer.component\_library.premarshaller.cursor.UrtCursorSerializer

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

/\*\*

\* Builds [[UrtOrderedCursor]] in the Bottom position

\*

\* @param idSelector Specifies the entry from which to derive the `id` field

\* @param includeOperation Logic to determine whether or not to build the bottom cursor, which only

\* applies if gap cursors are required (e.g. Home Latest). When applicable,

\* this logic should always be the inverse of the logic used to decide

\* whether or not to build the gap cursor via [[OrderedGapCursorBuilder]],

\* since either the gap or the bottom cursor must always be returned.

\* @param serializer Converts the cursor to an encoded string

\*/

case class OrderedBottomCursorBuilder[

-Query <: PipelineQuery with HasPipelineCursor[UrtOrderedCursor]

](

idSelector: PartialFunction[TimelineEntry, Long],

override val includeOperation: IncludeInstruction[Query] = AlwaysInclude,

serializer: PipelineCursorSerializer[UrtOrderedCursor] = UrtCursorSerializer)

extends UrtCursorBuilder[Query] {

override val cursorType: CursorType = BottomCursor

override def cursorValue(query: Query, timelineEntries: Seq[TimelineEntry]): String = {

val bottomId = timelineEntries.reverseIterator.collectFirst(idSelector)

val id = bottomId.orElse(query.pipelineCursor.flatMap(\_.id))

val cursor = UrtOrderedCursor(

initialSortIndex = nextBottomInitialSortIndex(query, timelineEntries),

id = id,

cursorType = Some(cursorType)

)

serializer.serializeCursor(cursor)

}

}