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

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

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

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

import com.twitter.product\_mixer.core.model.marshalling.response.slice.NextCursor

import com.twitter.product\_mixer.core.model.marshalling.response.slice.SliceItem

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 [[OrderedCursor]] in the Next position

\*

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

\* @param includeOperation Specifies whether to include the builder operation in the response

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

\*/

case class OrderedNextCursorBuilder[Query <: PipelineQuery with HasPipelineCursor[OrderedCursor]](

idSelector: PartialFunction[SliceItem, Long],

override val includeOperation: ShouldInclude[Query] = AlwaysInclude,

serializer: PipelineCursorSerializer[OrderedCursor] = CursorSerializer)

extends SliceCursorBuilder[Query] {

override val cursorType: CursorType = NextCursor

override def cursorValue(

query: Query,

entries: Seq[SliceItem]

): String = {

val bottomId = entries.reverseIterator.collectFirst(idSelector)

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

val cursor = OrderedCursor(id = id, cursorType = Some(cursorType))

serializer.serializeCursor(cursor)

}

}