package com.twitter.product\_mixer.component\_library.model.cursor

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

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

/\*\*

\* URT Cursor model that may be used when cursoring over a unordered candidate source. On each server

\* round-trip, the server will append the IDs of the elements in the response to the cursor. Then

\* on subsequent requests the client will return the cursor, and the excludedIds list can be sent to

\* the downstream's excludeIds parameter, or excluded locally via a filter on the candidate source

\* pipeline.

\*

\* Note that the cursor is bounded, as the excludedIds list cannot be appended to indefinitely due

\* to payload size constraints. As such, this strategy is typically used for bounded (limited page

\* size) products, or for unbounded (unlimited page size) products in conjunction with an

\* impression store. In the latter case, the cursor excludedIds list would be limited to a max size

\* via a circular buffer implementation, which would be unioned with the impression store IDs when

\* filtering. This usage allows the impression store to "catch up", as there is often latency

\* between when an impression client event is sent by the client and storage in the impression

\* store.

\*

\* @param initialSortIndex See [[UrtPipelineCursor]]

\* @param excludedIds the list of IDs to exclude from the candidate list

\*/

case class UrtUnorderedExcludeIdsCursor(

override val initialSortIndex: Long,

excludedIds: Seq[Long])

extends UrtPipelineCursor

case class UnorderedExcludeIdsCursor(excludedIds: Seq[Long]) extends PipelineCursor