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

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

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

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

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

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

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.PipelineCursorSerializer

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

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

object PlaceholderTopCursorBuilder {

val DefaultPlaceholderCursor = UrtPlaceholderCursor()

}

/\*\*

\* Top cursor builder that can be used when the Product does not support paging up. The URT spec

\* requires that both bottom and top cursors always be present on each page. Therefore, if the

\* product does not support paging up, then we can use a cursor value that is not deserializable.

\* This way if the client submits a TopCursor, the backend will treat the the request as if no

\* cursor was submitted.

\*/

case class PlaceholderTopCursorBuilder(

serializer: PipelineCursorSerializer[UrtPipelineCursor] = UrtCursorSerializer)

extends UrtCursorBuilder[PipelineQuery with HasPipelineCursor[UrtPipelineCursor]] {

override val cursorType: CursorType = TopCursor

override def cursorValue(

query: PipelineQuery with HasPipelineCursor[UrtPipelineCursor],

timelineEntries: Seq[TimelineEntry]

): String = serializer.serializeCursor(DefaultPlaceholderCursor)

}