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

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

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

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

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

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

/\*\*

\* Selects one or more [[TimelineEntry]] instance from the input timeline entries.

\*

\* @tparam Query The domain model for the [[PipelineQuery]] used as input.

\*/

trait EntriesToReplace[-Query <: PipelineQuery] {

def apply(query: Query, entries: Seq[TimelineEntry]): Seq[TimelineEntry]

}

/\*\*

\* Selects all entries with a non-empty valid entryIdToReplace.

\*

\* @note this will result in multiple [[ReplaceEntryTimelineInstruction]]s

\*/

case object ReplaceAllEntries extends EntriesToReplace[PipelineQuery] {

def apply(query: PipelineQuery, entries: Seq[TimelineEntry]): Seq[TimelineEntry] =

entries.filter(\_.entryIdToReplace.isDefined)

}

/\*\*

\* Selects a replaceable URT [[CursorOperation]] from the timeline entries, that matches the

\* input cursorType.

\*/

case class ReplaceUrtCursor(cursorType: CursorType) extends EntriesToReplace[PipelineQuery] {

override def apply(query: PipelineQuery, entries: Seq[TimelineEntry]): Seq[TimelineEntry] =

entries.collectFirst {

case cursorOperation: CursorOperation

if cursorOperation.cursorType == cursorType && cursorOperation.entryIdToReplace.isDefined =>

cursorOperation

}.toSeq

}

/\*\*

\* Create a ReplaceEntry instruction

\*

\* @param entriesToReplace each replace instruction can contain only one entry. Users specify which

\* entry to replace using [[EntriesToReplace]]. If multiple entries are

\* specified, multiple [[ReplaceEntryTimelineInstruction]]s will be created.

\* @param includeInstruction whether the instruction should be included in the response

\*/

case class ReplaceEntryInstructionBuilder[Query <: PipelineQuery](

entriesToReplace: EntriesToReplace[Query],

override val includeInstruction: IncludeInstruction[Query] = AlwaysInclude)

extends UrtInstructionBuilder[Query, ReplaceEntryTimelineInstruction] {

override def build(

query: Query,

entries: Seq[TimelineEntry]

): Seq[ReplaceEntryTimelineInstruction] = {

if (includeInstruction(query, entries))

entriesToReplace(query, entries).map(ReplaceEntryTimelineInstruction)

else

Seq.empty

}

}