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

import com.twitter.product\_mixer.component\_library.premarshaller.slice.builder.SliceCursorUpdater.getCursorByType

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

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

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

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

object SliceCursorUpdater {

def getCursorByType(

items: Seq[SliceItem],

cursorType: CursorType

): Option[CursorItem] = {

items.collectFirst {

case cursor: CursorItem if cursor.cursorType == cursorType => cursor

}

}

}

/\*\*

\* If [[SliceCursorBuilder.includeOperation]] is true and a cursor does exist in the `items`,

\* this will run the the underlying [[SliceCursorBuilder]] with the full `items`

\* (including all cursors which may be present) then filter out only the originally

\* found [[CursorItem]] from the results). Then append the new cursor to the end of the results.

\*

\* If you have multiple cursors that need to be updated, you will need to have multiple updaters.

\*

\* If a CursorCandidate is returned by a Candidate Source, use this trait to update the Cursor

\* (if necessary) and add it to the end of the candidates list.

\*/

trait SliceCursorUpdater[-Query <: PipelineQuery] extends SliceCursorBuilder[Query] { self =>

def getExistingCursor(items: Seq[SliceItem]): Option[CursorItem] = {

getCursorByType(items, self.cursorType)

}

def update(query: Query, items: Seq[SliceItem]): Seq[SliceItem] = {

if (includeOperation(query, items)) {

getExistingCursor(items)

.map { existingCursor =>

// Safe get because includeOperation() is shared in this context

val newCursor = build(query, items).get

items.filterNot(\_ == existingCursor) :+ newCursor

}.getOrElse(items)

} else items

}

}

trait SliceCursorUpdaterFromUnderlyingBuilder[-Query <: PipelineQuery]

extends SliceCursorUpdater[Query] {

def underlying: SliceCursorBuilder[Query]

override def cursorValue(

query: Query,

entries: Seq[SliceItem]

): String = underlying.cursorValue(query, entries)

}