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

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

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

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

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

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

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

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

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

import com.twitter.product\_mixer.core.pipeline.pipeline\_failure.PipelineFailure

import com.twitter.product\_mixer.core.pipeline.pipeline\_failure.UnexpectedCandidateInMarshaller

trait SliceBuilder[-Query <: PipelineQuery] {

def cursorBuilders: Seq[SliceCursorBuilder[Query]]

def cursorUpdaters: Seq[SliceCursorUpdater[Query]]

private def containsGapCursor(items: Seq[SliceItem]): Boolean =

items.collectFirst { case CursorItem(\_, GapCursor) => () }.nonEmpty

final def buildSlice(query: Query, items: Seq[SliceItem]): Slice = {

val builtCursors = cursorBuilders.flatMap(\_.build(query, items))

// Iterate over the cursorUpdaters in the order they were defined. Note that each updater will

// be passed the items updated by the previous cursorUpdater.

val updatedItems = cursorUpdaters.foldLeft(items) { (items, cursorUpdater) =>

cursorUpdater.update(query, items)

} ++ builtCursors

val (cursors, nonCursorItems) = updatedItems.partition(\_.isInstanceOf[CursorItem])

val nextCursor = cursors.collectFirst {

case cursor @ CursorItem(\_, NextCursor) => cursor.value

}

val previousCursor = cursors.collectFirst {

case cursor @ CursorItem(\_, PreviousCursor) => cursor.value

}

/\*\*

\* Identify whether a [[GapCursor]] is present and give as much detail to point to where it came from

\* Since this is already a fatal error case for the request, its okay to be a little expensive to get

\* the best error message possible for debug purposes.

\*/

if (containsGapCursor(cursors)) {

val errorDetails =

if (containsGapCursor(builtCursors)) {

"This means one of your `cursorBuilders` returned a GapCursor."

} else if (containsGapCursor(items)) {

"This means one of your `CandidateDecorator`s decorated a Candidate with a GapCursor."

} else {

"This means one of your `cursorUpdaters` returned a GapCursor."

}

throw PipelineFailure(

UnexpectedCandidateInMarshaller,

s"SliceBuilder does not support GapCursors but one was given. $errorDetails"

)

}

Slice(

items = nonCursorItems,

sliceInfo = SliceInfo(previousCursor = previousCursor, nextCursor = nextCursor))

}

}