package com.twitter.product\_mixer.core.functional\_component.marshaller.response.urt

import com.twitter.product\_mixer.core.functional\_component.marshaller.TransportMarshaller

import com.twitter.product\_mixer.core.functional\_component.marshaller.response.urt.metadata.ChildFeedbackActionMarshaller

import com.twitter.product\_mixer.core.functional\_component.marshaller.response.urt.metadata.FeedbackActionMarshaller

import com.twitter.product\_mixer.core.model.common.identifier.TransportMarshallerIdentifier

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

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

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

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

import com.twitter.timelines.render.thriftscala.TimelineResponse

import com.twitter.timelines.render.{thriftscala => urt}

import javax.inject.Inject

import javax.inject.Singleton

/\*\*

\* [[TransportMarshaller]] for URT types

\*

\* @note to make an instance of a [[UrtTransportMarshaller]] you can use [[UrtTransportMarshallerBuilder.marshaller]]

\*/

@Singleton

class UrtTransportMarshaller @Inject() (

timelineInstructionMarshaller: TimelineInstructionMarshaller,

feedbackActionMarshaller: FeedbackActionMarshaller,

childFeedbackActionMarshaller: ChildFeedbackActionMarshaller,

timelineMetadataMarshaller: TimelineMetadataMarshaller)

extends TransportMarshaller[Timeline, urt.TimelineResponse] {

override val identifier: TransportMarshallerIdentifier =

TransportMarshallerIdentifier("UnifiedRichTimeline")

override def apply(timeline: Timeline): urt.TimelineResponse = {

val feedbackActions: Option[Map[String, urt.FeedbackAction]] = {

collectAndMarshallFeedbackActions(timeline.instructions)

}

urt.TimelineResponse(

state = urt.TimelineState.Ok,

timeline = urt.Timeline(

id = timeline.id,

instructions = timeline.instructions.map(timelineInstructionMarshaller(\_)),

responseObjects =

feedbackActions.map(actions => urt.ResponseObjects(feedbackActions = Some(actions))),

metadata = timeline.metadata.map(timelineMetadataMarshaller(\_))

)

)

}

// Currently, feedbackActionInfo at the URT TimelineItem level is supported, which covers almost all

// existing use cases. However, if additional feedbackActionInfos are defined on the URT

// TimelineItemContent level for "compound" URT types (see deprecated TopicCollection /

// TopicCollectionData), this is not supported. If "compound" URT types are added in the future,

// support must be added within that type (see ModuleItem) to handle the collection and marshalling

// of these feedbackActionInfos.

private[this] def collectAndMarshallFeedbackActions(

instructions: Seq[TimelineInstruction]

): Option[Map[String, urt.FeedbackAction]] = {

val feedbackActions: Seq[FeedbackAction] = for {

feedbackActionInfos <- instructions.collect {

case c: ContainsFeedbackActionInfos => c.feedbackActionInfos

}

feedbackInfoOpt <- feedbackActionInfos

feedbackInfo <- feedbackInfoOpt.toSeq

feedbackAction <- feedbackInfo.feedbackActions

} yield feedbackAction

if (feedbackActions.nonEmpty) {

val urtFeedbackActions = feedbackActions.map(feedbackActionMarshaller(\_))

val urtChildFeedbackActions: Seq[urt.FeedbackAction] = for {

feedbackAction <- feedbackActions

childFeedbackActions <- feedbackAction.childFeedbackActions.toSeq

childFeedbackAction <- childFeedbackActions

} yield childFeedbackActionMarshaller(childFeedbackAction)

val allUrtFeedbackActions = urtFeedbackActions ++ urtChildFeedbackActions

Some(

allUrtFeedbackActions.map { urtAction =>

FeedbackActionMarshaller.generateKey(urtAction) -> urtAction

}.toMap

)

} else {

None

}

}

}

object UrtTransportMarshaller {

def unavailable(timelineId: String): TimelineResponse = {

urt.TimelineResponse(

state = urt.TimelineState.Unavailable,

timeline = urt.Timeline(

id = timelineId,

instructions = Seq.empty,

responseObjects = None,

metadata = None

)

)

}

}