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

import com.twitter.product\_mixer.component\_library.decorator.urt.builder.timeline\_module.ManualModuleId

import com.twitter.product\_mixer.component\_library.decorator.urt.builder.timeline\_module.ModuleIdGeneration

import com.twitter.product\_mixer.component\_library.decorator.urt.builder.timeline\_module.AutomaticUniqueModuleId

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

import com.twitter.product\_mixer.core.functional\_component.premarshaller.DomainMarshaller

import com.twitter.product\_mixer.core.functional\_component.premarshaller.UndecoratedCandidateDomainMarshallerException

import com.twitter.product\_mixer.core.functional\_component.premarshaller.UndecoratedModuleDomainMarshallerException

import com.twitter.product\_mixer.core.functional\_component.premarshaller.UnsupportedModuleDomainMarshallerException

import com.twitter.product\_mixer.core.functional\_component.premarshaller.UnsupportedPresentationDomainMarshallerException

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

import com.twitter.product\_mixer.core.model.common.presentation.CandidateWithDetails

import com.twitter.product\_mixer.core.model.common.presentation.ItemCandidateWithDetails

import com.twitter.product\_mixer.core.model.common.presentation.ModuleCandidateWithDetails

import com.twitter.product\_mixer.core.model.common.presentation.urt.BaseUrtItemPresentation

import com.twitter.product\_mixer.core.model.common.presentation.urt.BaseUrtModulePresentation

import com.twitter.product\_mixer.core.model.common.presentation.urt.BaseUrtOperationPresentation

import com.twitter.product\_mixer.core.model.common.presentation.urt.IsDispensable

import com.twitter.product\_mixer.core.model.common.presentation.urt.WithItemTreeDisplay

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

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.pipeline.PipelineQuery

/\*\*

\* Domain marshaller that generates URT timelines automatically if the candidate pipeline decorators

\* use item and module presentations types that implement [[BaseUrtItemPresentation]] and

\* [[BaseUrtModulePresentation]], respectively to hold URT presentation data.

\*/

case class UrtDomainMarshaller[-Query <: PipelineQuery](

override val instructionBuilders: Seq[UrtInstructionBuilder[Query, TimelineInstruction]] =

Seq(AddEntriesInstructionBuilder()),

override val cursorBuilders: Seq[UrtCursorBuilder[Query]] = Seq.empty,

override val cursorUpdaters: Seq[UrtCursorUpdater[Query]] = Seq.empty,

override val metadataBuilder: Option[BaseUrtMetadataBuilder[Query]] = None,

override val sortIndexStep: Int = 1,

override val identifier: DomainMarshallerIdentifier =

DomainMarshallerIdentifier("UnifiedRichTimeline"))

extends DomainMarshaller[Query, Timeline]

with UrtBuilder[Query, TimelineInstruction] {

override def apply(

query: Query,

selections: Seq[CandidateWithDetails]

): Timeline = {

val initialSortIndex = getInitialSortIndex(query)

val entries = selections.zipWithIndex.map {

case (ItemCandidateWithDetails(\_, Some(presentation: BaseUrtItemPresentation), \_), \_) =>

presentation.timelineItem

case (ItemCandidateWithDetails(\_, Some(presentation: BaseUrtOperationPresentation), \_), \_) =>

presentation.timelineOperation

case (

ModuleCandidateWithDetails(

candidates,

Some(presentation: BaseUrtModulePresentation),

\_),

index) =>

val moduleItems = candidates.collect {

case ItemCandidateWithDetails(\_, Some(itemPresentation: BaseUrtItemPresentation), \_) =>

buildModuleItem(itemPresentation)

}

ModuleIdGeneration(presentation.timelineModule.id) match {

case \_: AutomaticUniqueModuleId =>

// Module IDs are unique using this method since initialSortIndex is based on time of request combined

// with each timeline module index

presentation.timelineModule.copy(id = initialSortIndex + index, items = moduleItems)

case ManualModuleId(moduleId) =>

presentation.timelineModule.copy(id = moduleId, items = moduleItems)

}

case (

itemCandidateWithDetails @ ItemCandidateWithDetails(candidate, Some(presentation), \_),

\_) =>

throw new UnsupportedPresentationDomainMarshallerException(

candidate,

presentation,

itemCandidateWithDetails.source)

case (itemCandidateWithDetails @ ItemCandidateWithDetails(candidate, None, \_), \_) =>

throw new UndecoratedCandidateDomainMarshallerException(

candidate,

itemCandidateWithDetails.source)

case (

moduleCandidateWithDetails @ ModuleCandidateWithDetails(\_, presentation @ Some(\_), \_),

\_) =>

// handles given a non `BaseUrtModulePresentation` presentation type

throw new UnsupportedModuleDomainMarshallerException(

presentation,

moduleCandidateWithDetails.source)

case (moduleCandidateWithDetails @ ModuleCandidateWithDetails(\_, None, \_), \_) =>

throw new UndecoratedModuleDomainMarshallerException(moduleCandidateWithDetails.source)

}

buildTimeline(query, entries)

}

private def buildModuleItem(itemPresentation: BaseUrtItemPresentation): ModuleItem = {

val isDispensable = itemPresentation match {

case isDispensable: IsDispensable => Some(isDispensable.dispensable)

case \_ => None

}

val treeDisplay = itemPresentation match {

case withItemTreeDisplay: WithItemTreeDisplay => withItemTreeDisplay.treeDisplay

case \_ => None

}

ModuleItem(

itemPresentation.timelineItem,

dispensable = isDispensable,

treeDisplay = treeDisplay)

}

}