package com.twitter.product\_mixer.component\_library.decorator.urt.builder.item.audio\_space

import com.twitter.product\_mixer.component\_library.decorator.urt.builder.item.audio\_space.AudioSpaceCandidateUrtItemBuilder.AudioSpaceClientEventInfoElement

import com.twitter.product\_mixer.component\_library.model.candidate.AudioSpaceCandidate

import com.twitter.product\_mixer.core.feature.featuremap.FeatureMap

import com.twitter.product\_mixer.core.functional\_component.decorator.urt.builder.CandidateUrtEntryBuilder

import com.twitter.product\_mixer.core.functional\_component.decorator.urt.builder.metadata.BaseClientEventInfoBuilder

import com.twitter.product\_mixer.core.functional\_component.decorator.urt.builder.metadata.BaseFeedbackActionInfoBuilder

import com.twitter.product\_mixer.core.model.common.UniversalNoun

import com.twitter.product\_mixer.core.model.marshalling.response.urt.item.audio\_space.AudioSpaceItem

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

object AudioSpaceCandidateUrtItemBuilder {

val AudioSpaceClientEventInfoElement: String = "audiospace"

}

case class AudioSpaceCandidateUrtItemBuilder[-Query <: PipelineQuery](

clientEventInfoBuilder: BaseClientEventInfoBuilder[Query, UniversalNoun[Any]],

feedbackActionInfoBuilder: Option[

BaseFeedbackActionInfoBuilder[Query, UniversalNoun[Any]]

] = None)

extends CandidateUrtEntryBuilder[Query, AudioSpaceCandidate, AudioSpaceItem] {

override def apply(

query: Query,

audioSpaceCandidate: AudioSpaceCandidate,

candidateFeatures: FeatureMap

): AudioSpaceItem = AudioSpaceItem(

id = audioSpaceCandidate.id,

sortIndex = None, // Sort indexes are automatically set in the domain marshaller phase

clientEventInfo = clientEventInfoBuilder(

query,

audioSpaceCandidate,

candidateFeatures,

Some(AudioSpaceClientEventInfoElement)),

feedbackActionInfo =

feedbackActionInfoBuilder.flatMap(\_.apply(query, audioSpaceCandidate, candidateFeatures))

)

}