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

import com.twitter.product\_mixer.component\_library.decorator.urt.builder.item.topic.TopicCandidateUrtItemBuilder.TopicClientEventInfoElement

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

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.item.topic.BaseTopicDisplayTypeBuilder

import com.twitter.product\_mixer.core.functional\_component.decorator.urt.builder.item.topic.BaseTopicFunctionalityTypeBuilder

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.marshalling.response.urt.item.topic.TopicItem

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

object TopicCandidateUrtItemBuilder {

val TopicClientEventInfoElement: String = "topic"

}

case class TopicCandidateUrtItemBuilder[-Query <: PipelineQuery, Candidate <: BaseTopicCandidate](

clientEventInfoBuilder: BaseClientEventInfoBuilder[Query, Candidate],

topicFunctionalityTypeBuilder: Option[BaseTopicFunctionalityTypeBuilder[Query, Candidate]] = None,

topicDisplayTypeBuilder: Option[BaseTopicDisplayTypeBuilder[Query, Candidate]] = None,

feedbackActionInfoBuilder: Option[

BaseFeedbackActionInfoBuilder[Query, Candidate]

] = None)

extends CandidateUrtEntryBuilder[Query, Candidate, TopicItem] {

override def apply(

query: Query,

topicCandidate: Candidate,

candidateFeatures: FeatureMap

): TopicItem =

TopicItem(

id = topicCandidate.id,

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

clientEventInfo = clientEventInfoBuilder(

query,

topicCandidate,

candidateFeatures,

Some(TopicClientEventInfoElement)),

feedbackActionInfo =

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

topicFunctionalityType =

topicFunctionalityTypeBuilder.flatMap(\_.apply(query, topicCandidate, candidateFeatures)),

topicDisplayType =

topicDisplayTypeBuilder.flatMap(\_.apply(query, topicCandidate, candidateFeatures))

)

}