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

import com.twitter.product\_mixer.component\_library.decorator.urt.builder.item.commerce.CommerceProductCandidateUrtItemBuilder.CommerceProductClientEventInfoElement

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

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.marshalling.response.urt.item.commerce.CommerceProductItem

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

object CommerceProductCandidateUrtItemBuilder {

val CommerceProductClientEventInfoElement: String = "commerce-product"

}

case class CommerceProductCandidateUrtItemBuilder[-Query <: PipelineQuery](

clientEventInfoBuilder: BaseClientEventInfoBuilder[Query, CommerceProductCandidate],

feedbackActionInfoBuilder: Option[BaseFeedbackActionInfoBuilder[Query, CommerceProductCandidate]])

extends CandidateUrtEntryBuilder[

Query,

CommerceProductCandidate,

CommerceProductItem

] {

override def apply(

query: Query,

candidate: CommerceProductCandidate,

candidateFeatures: FeatureMap

): CommerceProductItem =

CommerceProductItem(

id = candidate.id,

sortIndex = None,

clientEventInfo = clientEventInfoBuilder(

query,

candidate,

candidateFeatures,

Some(CommerceProductClientEventInfoElement)),

feedbackActionInfo =

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

)

}