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

import com.twitter.product\_mixer.component\_library.decorator.urt.builder.item.commerce.CommerceProductGroupCandidateUrtItemBuilder.CommerceProductGroupClientEventInfoElement

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

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.CommerceProductGroupItem

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

object CommerceProductGroupCandidateUrtItemBuilder {

val CommerceProductGroupClientEventInfoElement: String = "commerce-product-group"

}

case class CommerceProductGroupCandidateUrtItemBuilder[-Query <: PipelineQuery](

clientEventInfoBuilder: BaseClientEventInfoBuilder[Query, CommerceProductGroupCandidate],

feedbackActionInfoBuilder: Option[

BaseFeedbackActionInfoBuilder[Query, CommerceProductGroupCandidate]

]) extends CandidateUrtEntryBuilder[

Query,

CommerceProductGroupCandidate,

CommerceProductGroupItem

] {

override def apply(

query: Query,

candidate: CommerceProductGroupCandidate,

candidateFeatures: FeatureMap

): CommerceProductGroupItem =

CommerceProductGroupItem(

id = candidate.id,

sortIndex = None,

clientEventInfo = clientEventInfoBuilder(

query,

candidate,

candidateFeatures,

Some(CommerceProductGroupClientEventInfoElement)),

feedbackActionInfo =

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

)

}