package com.twitter.product\_mixer.component\_library.pipeline.candidate.ads

import com.twitter.adserver.{thriftscala => ads}

import com.twitter.product\_mixer.component\_library.model.query.ads.AdsQuery

import com.twitter.product\_mixer.component\_library.pipeline.candidate.ads.AdsCandidatePipelineQueryTransformer.buildAdRequestParams

import com.twitter.product\_mixer.core.functional\_component.transformer.DependentCandidatePipelineQueryTransformer

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

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

/\*\*

\* Transform a PipelineQuery with AdsQuery into an AdsRequestParams

\*

\* @param adsDisplayLocationBuilder Builder that determines the display location for the ads

\* @param countNumOrganicItems Count organic items from the response

\*/

case class AdsDependentCandidatePipelineQueryTransformer[Query <: PipelineQuery with AdsQuery](

adsDisplayLocationBuilder: AdsDisplayLocationBuilder[Query],

getOrganicItemIds: GetOrganicItemIds,

countNumOrganicItems: CountNumOrganicItems[Query],

urtRequest: Option[Boolean],

) extends DependentCandidatePipelineQueryTransformer[Query, ads.AdRequestParams] {

override def transform(

query: Query,

previousCandidates: Seq[CandidateWithDetails]

): ads.AdRequestParams = buildAdRequestParams(

query = query,

adsDisplayLocation = adsDisplayLocationBuilder(query),

organicItemIds = getOrganicItemIds.apply(previousCandidates),

numOrganicItems = Some(countNumOrganicItems.apply(query, previousCandidates)),

urtRequest = urtRequest

)

}