package com.twitter.home\_mixer.product.for\_you.side\_effect

import com.twitter.home\_mixer.model.HomeFeatures.IsReadFromCacheFeature

import com.twitter.home\_mixer.model.HomeFeatures.PredictionRequestIdFeature

import com.twitter.home\_mixer.model.HomeFeatures.ServedIdFeature

import com.twitter.home\_mixer.model.HomeFeatures.ServedRequestIdFeature

import com.twitter.home\_mixer.model.HomeFeatures.StreamToKafkaFeature

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

import com.twitter.product\_mixer.core.model.common.identifier.CandidatePipelineIdentifier

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

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

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

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

object ServedCandidateKafkaSideEffect {

def extractCandidates(

query: PipelineQuery,

selectedCandidates: Seq[CandidateWithDetails],

sourceIdentifiers: Set[CandidatePipelineIdentifier]

): Seq[ItemCandidateWithDetails] = {

val servedRequestIdOpt =

query.features.getOrElse(FeatureMap.empty).getOrElse(ServedRequestIdFeature, None)

selectedCandidates.iterator

.filter(candidate => sourceIdentifiers.contains(candidate.source))

.flatMap {

case item: ItemCandidateWithDetails => Seq(item)

case module: ModuleCandidateWithDetails => module.candidates

}

.filter(candidate => candidate.features.getOrElse(StreamToKafkaFeature, false))

.map { candidate =>

val servedId =

if (candidate.features.getOrElse(IsReadFromCacheFeature, false) &&

servedRequestIdOpt.nonEmpty)

servedRequestIdOpt

else

candidate.features.getOrElse(PredictionRequestIdFeature, None)

candidate.copy(features = candidate.features + (ServedIdFeature, servedId))

}.toSeq

// deduplicate by (tweetId, userId, servedId)

.groupBy { candidate =>

(

candidate.candidateIdLong,

query.getRequiredUserId,

candidate.features.getOrElse(ServedIdFeature, None))

}.values.map(\_.head).toSeq

}

}