package com.twitter.follow\_recommendations.common.feature\_hydration.sources

import com.google.inject.Provides

import com.google.inject.Singleton

import com.twitter.follow\_recommendations.common.feature\_hydration.adapters.ClientContextAdapter

import com.twitter.follow\_recommendations.common.feature\_hydration.common.FeatureSource

import com.twitter.follow\_recommendations.common.feature\_hydration.common.FeatureSourceId

import com.twitter.follow\_recommendations.common.feature\_hydration.common.HasPreFetchedFeature

import com.twitter.follow\_recommendations.common.models.CandidateUser

import com.twitter.follow\_recommendations.common.models.HasDisplayLocation

import com.twitter.follow\_recommendations.common.models.HasSimilarToContext

import com.twitter.ml.api.DataRecord

import com.twitter.ml.api.FeatureContext

import com.twitter.product\_mixer.core.model.marshalling.request.HasClientContext

import com.twitter.stitch.Stitch

import com.twitter.timelines.configapi.HasParams

/\*\*

\* This source only takes features from the request (e.g. client context, WTF display location)

\* No external calls are made.

\*/

@Provides

@Singleton

class ClientContextSource() extends FeatureSource {

override val id: FeatureSourceId = FeatureSourceId.ClientContextSourceId

override val featureContext: FeatureContext = ClientContextAdapter.getFeatureContext

override def hydrateFeatures(

t: HasClientContext

with HasPreFetchedFeature

with HasParams

with HasSimilarToContext

with HasDisplayLocation,

candidates: Seq[CandidateUser]

): Stitch[Map[CandidateUser, DataRecord]] = {

Stitch.value(

candidates

.map(\_ -> ((t.clientContext, t.displayLocation))).toMap.mapValues(

ClientContextAdapter.adaptToDataRecord))

}

}