package com.twitter.home\_mixer.product.scored\_tweets.feature\_hydrator.real\_time\_aggregates

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

import com.twitter.product\_mixer.core.feature.Feature

import com.twitter.product\_mixer.core.feature.datarecord.DataRecordInAFeature

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

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

import com.twitter.product\_mixer.core.functional\_component.feature\_hydrator.BulkCandidateFeatureHydrator

import com.twitter.product\_mixer.core.model.common.CandidateWithFeatures

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

import com.twitter.product\_mixer.core.util.OffloadFuturePools

import com.twitter.stitch.Stitch

trait BaseRealTimeAggregateBulkCandidateFeatureHydrator[K]

extends BulkCandidateFeatureHydrator[PipelineQuery, TweetCandidate]

with BaseRealtimeAggregateHydrator[K] {

val outputFeature: DataRecordInAFeature[TweetCandidate]

override def features: Set[Feature[\_, \_]] = Set(outputFeature)

override lazy val statScope: String = identifier.toString

def keysFromQueryAndCandidates(

query: PipelineQuery,

candidates: Seq[CandidateWithFeatures[TweetCandidate]]

): Seq[Option[K]]

override def apply(

query: PipelineQuery,

candidates: Seq[CandidateWithFeatures[TweetCandidate]]

): Stitch[Seq[FeatureMap]] = OffloadFuturePools.offloadFuture {

val possiblyKeys = keysFromQueryAndCandidates(query, candidates)

fetchAndConstructDataRecords(possiblyKeys).map { dataRecords =>

dataRecords.map { dataRecord =>

FeatureMapBuilder().add(outputFeature, dataRecord).build()

}

}

}

}