package com.twitter.home\_mixer.product.scored\_tweets.query\_transformer.earlybird

import com.twitter.conversions.DurationOps.\_

import com.twitter.home\_mixer.model.request.HasDeviceContext

import com.twitter.home\_mixer.product.scored\_tweets.feature\_hydrator.FrsSeedUserIdsFeature

import com.twitter.home\_mixer.product.scored\_tweets.query\_transformer.earlybird.EarlybirdFrsQueryTransformer.\_

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

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

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

import com.twitter.product\_mixer.core.quality\_factor.HasQualityFactorStatus

import com.twitter.search.earlybird.{thriftscala => eb}

import com.twitter.timelines.common.model.TweetKindOption

object EarlybirdFrsQueryTransformer {

private val SinceDuration = 24.hours

private val MaxTweetsToFetch = 100

private val TensorflowModel = Some("timelines\_rectweet\_replica")

private val TweetKindOptions: TweetKindOption.ValueSet =

TweetKindOption(includeOriginalTweetsAndQuotes = true)

}

case class EarlybirdFrsQueryTransformer[

Query <: PipelineQuery with HasQualityFactorStatus with HasDeviceContext

](

candidatePipelineIdentifier: CandidatePipelineIdentifier,

override val clientId: Option[String])

extends CandidatePipelineQueryTransformer[Query, eb.EarlybirdRequest]

with EarlybirdQueryTransformer[Query] {

override val tweetKindOptions: TweetKindOption.ValueSet = TweetKindOptions

override val maxTweetsToFetch: Int = MaxTweetsToFetch

override val tensorflowModel: Option[String] = TensorflowModel

override def transform(query: Query): eb.EarlybirdRequest = {

val seedUserIds = query.features

.flatMap(\_.getOrElse(FrsSeedUserIdsFeature, None))

.getOrElse(Seq.empty).toSet

buildEarlybirdQuery(query, SinceDuration, seedUserIds)

}

}