package com.twitter.home\_mixer.product.scored\_tweets.model

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

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

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

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

import com.twitter.product\_mixer.component\_library.model.cursor.UrtOrderedCursor

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

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

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

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

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

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

import com.twitter.timelines.configapi.Params

case class ScoredTweetsQuery(

override val params: Params,

override val clientContext: ClientContext,

override val pipelineCursor: Option[UrtOrderedCursor],

override val requestedMaxResults: Option[Int],

override val debugOptions: Option[DebugOptions],

override val features: Option[FeatureMap],

override val deviceContext: Option[DeviceContext],

override val seenTweetIds: Option[Seq[Long]],

override val qualityFactorStatus: Option[QualityFactorStatus])

extends PipelineQuery

with HasPipelineCursor[UrtOrderedCursor]

with HasDeviceContext

with HasSeenTweetIds

with HasQualityFactorStatus {

override val product: Product = ScoredTweetsProduct

override def withFeatureMap(features: FeatureMap): ScoredTweetsQuery =

copy(features = Some(features))

override def withQualityFactorStatus(

qualityFactorStatus: QualityFactorStatus

): ScoredTweetsQuery = copy(qualityFactorStatus = Some(qualityFactorStatus))

}