package com.twitter.product\_mixer.core.pipeline

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

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

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

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

import com.twitter.timelines.configapi.HasParams

import com.twitter.timelines.configapi.Param

import com.twitter.util.Time

trait PipelineQuery extends HasParams with HasClientContext with HasProduct with HasDebugOptions {

self =>

/\*\* Set a query time val that is constant for the duration of the query lifecycle \*/

val queryTime: Time = self.debugOptions.flatMap(\_.requestTimeOverride).getOrElse(Time.now)

/\*\* The requested max results is specified, or not specified, by the thrift client \*/

def requestedMaxResults: Option[Int]

/\*\* Retrieves the max results with a default Param, if not specified by the thrift client \*/

def maxResults(defaultRequestedMaxResultParam: Param[Int]): Int =

requestedMaxResults.getOrElse(params(defaultRequestedMaxResultParam))

/\*\* Optional [[FeatureMap]], this may be updated later using [[withFeatureMap]] \*/

def features: Option[FeatureMap]

/\*\*

\* Since Query-Level features can be hydrated later, we need this method to update the PipelineQuery

\* usually this will be implemented via `copy(features = Some(features))`

\*/

def withFeatureMap(features: FeatureMap): PipelineQuery

}