package com.twitter.timelineranker.parameters.revchron

import com.twitter.timelineranker.model.ReverseChronTimelineQuery

import com.twitter.timelines.util.bounds.BoundsWithDefault

import com.twitter.timelineservice.model.core.TimelineKind

import com.twitter.timelineservice.model.core.TimelineLimits

object ReverseChronTimelineQueryContext {

val MaxCountLimit: Int = TimelineLimits.default.lengthLimit(TimelineKind.home)

val MaxCount: BoundsWithDefault[Int] = BoundsWithDefault[Int](0, MaxCountLimit, MaxCountLimit)

val MaxCountMultiplier: BoundsWithDefault[Double] = BoundsWithDefault[Double](0.5, 2.0, 1.0)

val MaxFollowedUsers: BoundsWithDefault[Int] = BoundsWithDefault[Int](1, 15000, 5000)

val TweetsFilteringLossageThresholdPercent: BoundsWithDefault[Int] =

BoundsWithDefault[Int](10, 100, 20)

val TweetsFilteringLossageLimitPercent: BoundsWithDefault[Int] =

BoundsWithDefault[Int](40, 65, 60)

def getDefaultContext(query: ReverseChronTimelineQuery): ReverseChronTimelineQueryContext = {

new ReverseChronTimelineQueryContextImpl(

query,

getMaxCount = () => MaxCount.default,

getMaxCountMultiplier = () => MaxCountMultiplier.default,

getMaxFollowedUsers = () => MaxFollowedUsers.default,

getReturnEmptyWhenOverMaxFollows = () => true,

getDirectedAtNarrowastingViaSearch = () => false,

getPostFilteringBasedOnSearchMetadataEnabled = () => true,

getBackfillFilteredEntries = () => false,

getTweetsFilteringLossageThresholdPercent = () =>

TweetsFilteringLossageThresholdPercent.default,

getTweetsFilteringLossageLimitPercent = () => TweetsFilteringLossageLimitPercent.default

)

}

}

// Note that methods that return parameter value always use () to indicate that

// side effects may be involved in their invocation.

// for example, A likely side effect is to cause experiment impression.

trait ReverseChronTimelineQueryContext {

def query: ReverseChronTimelineQuery

// Maximum number of tweets to be returned to caller.

def maxCount(): Int

// Multiplier applied to the number of tweets fetched from search expressed as percentage.

// It can be used to fetch more than the number tweets requested by a caller (to improve similarity)

// or to fetch less than requested to reduce load.

def maxCountMultiplier(): Double

// Maximum number of followed user accounts to use when materializing home timelines.

def maxFollowedUsers(): Int

// When true, if the user follows more than maxFollowedUsers, return an empty timeline.

def returnEmptyWhenOverMaxFollows(): Boolean

// When true, appends an operator for directed-at narrowcasting to the home materialization

// search request

def directedAtNarrowcastingViaSearch(): Boolean

// When true, requests additional metadata from search and use this metadata for post filtering.

def postFilteringBasedOnSearchMetadataEnabled(): Boolean

// Controls whether to back-fill timeline entries that get filtered out by TweetsPostFilter

// during home timeline materialization.

def backfillFilteredEntries(): Boolean

// If back-filling filtered entries is enabled and if number of tweets that get filtered out

// exceed this percentage then we will issue a second call to get more tweets.

def tweetsFilteringLossageThresholdPercent(): Int

// We need to ensure that the number of tweets requested by the second call

// are not unbounded (for example, if everything is filtered out in the first call)

// therefore we adjust the actual filtered out percentage to be no greater than

// the value below.

def tweetsFilteringLossageLimitPercent(): Int

// We need to indicate to search if we should use the archive cluster

// this option will come from ReverseChronTimelineQueryOptions and

// will be `true` by default if the options are not present.

def getTweetsFromArchiveIndex(): Boolean =

query.options.map(\_.getTweetsFromArchiveIndex).getOrElse(true)

}

class ReverseChronTimelineQueryContextImpl(

override val query: ReverseChronTimelineQuery,

getMaxCount: () => Int,

getMaxCountMultiplier: () => Double,

getMaxFollowedUsers: () => Int,

getReturnEmptyWhenOverMaxFollows: () => Boolean,

getDirectedAtNarrowastingViaSearch: () => Boolean,

getPostFilteringBasedOnSearchMetadataEnabled: () => Boolean,

getBackfillFilteredEntries: () => Boolean,

getTweetsFilteringLossageThresholdPercent: () => Int,

getTweetsFilteringLossageLimitPercent: () => Int)

extends ReverseChronTimelineQueryContext {

override def maxCount(): Int = { getMaxCount() }

override def maxCountMultiplier(): Double = { getMaxCountMultiplier() }

override def maxFollowedUsers(): Int = { getMaxFollowedUsers() }

override def backfillFilteredEntries(): Boolean = { getBackfillFilteredEntries() }

override def tweetsFilteringLossageThresholdPercent(): Int = {

getTweetsFilteringLossageThresholdPercent()

}

override def tweetsFilteringLossageLimitPercent(): Int = {

getTweetsFilteringLossageLimitPercent()

}

override def returnEmptyWhenOverMaxFollows(): Boolean = {

getReturnEmptyWhenOverMaxFollows()

}

override def directedAtNarrowcastingViaSearch(): Boolean = {

getDirectedAtNarrowastingViaSearch()

}

override def postFilteringBasedOnSearchMetadataEnabled(): Boolean = {

getPostFilteringBasedOnSearchMetadataEnabled()

}

}