package com.twitter.timelineranker.parameters.revchron

import com.twitter.timelineranker.config.RuntimeConfiguration

import com.twitter.timelineranker.decider.DeciderKey

import com.twitter.timelineranker.model.\_

import com.twitter.timelineranker.parameters.util.RequestContextBuilder

import com.twitter.timelines.configapi.Config

import com.twitter.timelines.decider.FeatureValue

import com.twitter.util.Future

object ReverseChronTimelineQueryContextBuilder {

val MaxCountLimitKey: Seq[String] = Seq("search\_request\_max\_count\_limit")

}

class ReverseChronTimelineQueryContextBuilder(

config: Config,

runtimeConfig: RuntimeConfiguration,

requestContextBuilder: RequestContextBuilder) {

import ReverseChronTimelineQueryContext.\_

import ReverseChronTimelineQueryContextBuilder.\_

private val maxCountMultiplier = FeatureValue(

runtimeConfig.deciderGateBuilder,

DeciderKey.MultiplierOfMaterializationTweetsFetched,

MaxCountMultiplier,

value => (value / 100.0)

)

private val backfillFilteredEntriesGate =

runtimeConfig.deciderGateBuilder.linearGate(DeciderKey.BackfillFilteredEntries)

private val tweetsFilteringLossageThresholdPercent = FeatureValue(

runtimeConfig.deciderGateBuilder,

DeciderKey.TweetsFilteringLossageThreshold,

TweetsFilteringLossageThresholdPercent,

value => (value / 100)

)

private val tweetsFilteringLossageLimitPercent = FeatureValue(

runtimeConfig.deciderGateBuilder,

DeciderKey.TweetsFilteringLossageLimit,

TweetsFilteringLossageLimitPercent,

value => (value / 100)

)

private def getMaxCountFromConfigStore(): Int = {

runtimeConfig.configStore.getAsInt(MaxCountLimitKey).getOrElse(MaxCount.default)

}

def apply(query: ReverseChronTimelineQuery): Future[ReverseChronTimelineQueryContext] = {

requestContextBuilder(Some(query.userId), deviceContext = None).map { baseContext =>

val params = config(baseContext, runtimeConfig.statsReceiver)

new ReverseChronTimelineQueryContextImpl(

query,

getMaxCount = () => getMaxCountFromConfigStore(),

getMaxCountMultiplier = () => maxCountMultiplier(),

getMaxFollowedUsers = () => params(ReverseChronParams.MaxFollowedUsersParam),

getReturnEmptyWhenOverMaxFollows =

() => params(ReverseChronParams.ReturnEmptyWhenOverMaxFollowsParam),

getDirectedAtNarrowastingViaSearch =

() => params(ReverseChronParams.DirectedAtNarrowcastingViaSearchParam),

getPostFilteringBasedOnSearchMetadataEnabled =

() => params(ReverseChronParams.PostFilteringBasedOnSearchMetadataEnabledParam),

getBackfillFilteredEntries = () => backfillFilteredEntriesGate(),

getTweetsFilteringLossageThresholdPercent = () => tweetsFilteringLossageThresholdPercent(),

getTweetsFilteringLossageLimitPercent = () => tweetsFilteringLossageLimitPercent()

)

}

}

}