package com.twitter.visibility.configapi

import com.twitter.abdecider.LoggingABDecider

import com.twitter.decider.Decider

import com.twitter.featureswitches.v2.FeatureSwitches

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.logging.Logger

import com.twitter.servo.util.MemoizingStatsReceiver

import com.twitter.timelines.configapi.Params

import com.twitter.visibility.models.SafetyLevel

import com.twitter.visibility.models.UnitOfDiversion

import com.twitter.visibility.models.ViewerContext

object VisibilityParams {

def apply(

log: Logger,

statsReceiver: StatsReceiver,

decider: Decider,

abDecider: LoggingABDecider,

featureSwitches: FeatureSwitches

): VisibilityParams =

new VisibilityParams(log, statsReceiver, decider, abDecider, featureSwitches)

}

class VisibilityParams(

log: Logger,

statsReceiver: StatsReceiver,

decider: Decider,

abDecider: LoggingABDecider,

featureSwitches: FeatureSwitches) {

private[this] val contextFactory = new VisibilityRequestContextFactory(

abDecider,

featureSwitches

)

private[this] val configBuilder = ConfigBuilder(statsReceiver.scope("config"), decider, log)

private[this] val paramStats: MemoizingStatsReceiver = new MemoizingStatsReceiver(

statsReceiver.scope("configapi\_params"))

def apply(

viewerContext: ViewerContext,

safetyLevel: SafetyLevel,

unitsOfDiversion: Seq[UnitOfDiversion] = Seq.empty

): Params = {

val config = configBuilder.build(safetyLevel)

val requestContext = contextFactory(viewerContext, safetyLevel, unitsOfDiversion)

config.apply(requestContext, paramStats)

}

def memoized(

viewerContext: ViewerContext,

safetyLevel: SafetyLevel,

unitsOfDiversion: Seq[UnitOfDiversion] = Seq.empty

): Params = {

val config = configBuilder.buildMemoized(safetyLevel)

val requestContext = contextFactory(viewerContext, safetyLevel, unitsOfDiversion)

config.apply(requestContext, paramStats)

}

}