package com.twitter.visibility.configapi.configs

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.logging.Logger

import com.twitter.timelines.configapi.\_

import com.twitter.util.Time

import com.twitter.visibility.configapi.params.FSEnumRuleParam

import com.twitter.visibility.configapi.params.FSRuleParams.\_

private[visibility] object VisibilityFeatureSwitches {

val booleanFsOverrides: Seq[OptionalOverride[Boolean]] =

FeatureSwitchOverrideUtil.getBooleanFSOverrides(

AgeGatingAdultContentExperimentRuleEnabledParam,

CommunityTweetCommunityUnavailableLimitedActionsRulesEnabledParam,

CommunityTweetDropProtectedRuleEnabledParam,

CommunityTweetDropRuleEnabledParam,

CommunityTweetLimitedActionsRulesEnabledParam,

CommunityTweetMemberRemovedLimitedActionsRulesEnabledParam,

CommunityTweetNonMemberLimitedActionsRuleEnabledParam,

NsfwAgeBasedDropRulesHoldbackParam,

SkipTweetDetailLimitedEngagementRuleEnabledParam,

StaleTweetLimitedActionsRulesEnabledParam,

TrustedFriendsTweetLimitedEngagementsRuleEnabledParam,

FosnrFallbackDropRulesEnabledParam,

FosnrRulesEnabledParam

)

val doubleFsOverrides: Seq[OptionalOverride[Double]] =

FeatureSwitchOverrideUtil.getBoundedDoubleFSOverrides(

HighSpammyTweetContentScoreSearchTopProdTweetLabelDropRuleThresholdParam,

HighSpammyTweetContentScoreSearchLatestProdTweetLabelDropRuleThresholdParam,

HighSpammyTweetContentScoreTrendTopTweetLabelDropRuleThresholdParam,

HighSpammyTweetContentScoreTrendLatestTweetLabelDropRuleThresholdParam,

HighSpammyTweetContentScoreConvoDownrankAbusiveQualityThresholdParam,

HighToxicityModelScoreSpaceThresholdParam,

AdAvoidanceHighToxicityModelScoreThresholdParam,

AdAvoidanceReportedTweetModelScoreThresholdParam,

)

val timeFsOverrides: Seq[OptionalOverride[Time]] =

FeatureSwitchOverrideUtil.getTimeFromStringFSOverrides()

val stringSeqFeatureSwitchOverrides: Seq[OptionalOverride[Seq[String]]] =

FeatureSwitchOverrideUtil.getStringSeqFSOverrides(

CountrySpecificNsfwContentGatingCountriesParam,

AgeGatingAdultContentExperimentCountriesParam,

CardUriRootDomainDenyListParam

)

val enumFsParams: Seq[FSEnumRuleParam[\_ <: Enumeration]] = Seq()

val mkOptionalEnumFsOverrides: (StatsReceiver, Logger) => Seq[OptionalOverride[\_]] = {

(statsReceiver: StatsReceiver, logger: Logger) =>

FeatureSwitchOverrideUtil.getEnumFSOverrides(

statsReceiver,

logger,

enumFsParams: \_\*

)

}

def overrides(statsReceiver: StatsReceiver, logger: Logger): Seq[OptionalOverride[\_]] = {

val enumOverrides = mkOptionalEnumFsOverrides(statsReceiver, logger)

booleanFsOverrides ++

doubleFsOverrides ++

timeFsOverrides ++

stringSeqFeatureSwitchOverrides ++

enumOverrides

}

def config(statsReceiver: StatsReceiver, logger: Logger): BaseConfig =

BaseConfigBuilder(overrides(statsReceiver.scope("features\_switches"), logger))

.build("VisibilityFeatureSwitches")

}