package com.twitter.visibility.configapi

import com.twitter.abdecider.LoggingABDecider

import com.twitter.featureswitches.FSRecipient

import com.twitter.featureswitches.v2.FeatureSwitches

import com.twitter.timelines.configapi.abdecider.UserRecipientExperimentContextFactory

import com.twitter.timelines.configapi.featureswitches.v2.FeatureSwitchResultsFeatureContext

import com.twitter.timelines.configapi.FeatureContext

import com.twitter.timelines.configapi.NullExperimentContext

import com.twitter.timelines.configapi.UseFeatureContextExperimentContext

import com.twitter.visibility.models.SafetyLevel

import com.twitter.visibility.models.UnitOfDiversion

import com.twitter.visibility.models.ViewerContext

class VisibilityRequestContextFactory(

loggingABDecider: LoggingABDecider,

featureSwitches: FeatureSwitches) {

private val userExperimentContextFactory = new UserRecipientExperimentContextFactory(

loggingABDecider

)

private[this] def getFeatureContext(

context: ViewerContext,

safetyLevel: SafetyLevel,

unitsOfDiversion: Seq[UnitOfDiversion]

): FeatureContext = {

val uodCustomFields = unitsOfDiversion.map(\_.apply)

val recipient = FSRecipient(

userId = context.userId,

guestId = context.guestId,

userAgent = context.fsUserAgent,

clientApplicationId = context.clientApplicationId,

countryCode = context.requestCountryCode,

deviceId = context.deviceId,

languageCode = context.requestLanguageCode,

isTwoffice = Some(context.isTwOffice),

userRoles = context.userRoles,

).withCustomFields(("safety\_level", safetyLevel.name), uodCustomFields: \_\*)

val results = featureSwitches.matchRecipient(recipient)

new FeatureSwitchResultsFeatureContext(results)

}

def apply(

context: ViewerContext,

safetyLevel: SafetyLevel,

unitsOfDiversion: Seq[UnitOfDiversion] = Seq.empty

): VisibilityRequestContext = {

val experimentContextBase =

context.userId

.map(userId => userExperimentContextFactory.apply(userId)).getOrElse(NullExperimentContext)

val featureContext = getFeatureContext(context, safetyLevel, unitsOfDiversion)

val experimentContext =

UseFeatureContextExperimentContext(experimentContextBase, featureContext)

VisibilityRequestContext(

context.userId,

context.guestId,

experimentContext,

featureContext

)

}

}