package com.twitter.product\_mixer.core.functional\_component.configapi

import com.twitter.featureswitches.v2.FeatureSwitches

import com.twitter.featureswitches.UserAgent

import com.twitter.featureswitches.{Recipient => FeatureSwitchRecipient}

import com.twitter.product\_mixer.core.model.marshalling.request.ClientContext

import com.twitter.product\_mixer.core.model.marshalling.request.Product

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

import com.twitter.timelines.configapi.FeatureContext

import com.twitter.timelines.configapi.FeatureValue

import com.twitter.timelines.configapi.ForcedFeatureContext

import com.twitter.timelines.configapi.OrElseFeatureContext

import javax.inject.Inject

import javax.inject.Singleton

/\*\*

\* Request Context Factory is used to build RequestContext objects which are used

\* by the config api to determine the param overrides to apply to the request.

\* The param overrides are determined per request by configs which specify which

\* FS/Deciders/AB translate to what param overrides.

\*/

@Singleton

class RequestContextBuilder @Inject() (featureSwitches: FeatureSwitches) {

/\*\*

\* @param `fsCustomMapInput` allows you to set custom fields on your feature switches.

\* This feature isn't directly supported by product mixer yet, so using this argument

\* will likely result in future cleanup work.

\*

\*/

def build(

clientContext: ClientContext,

product: Product,

featureOverrides: Map[String, FeatureValue],

fsCustomMapInput: Map[String, Any]

): RequestContext = {

val featureContext =

getFeatureContext(clientContext, product, featureOverrides, fsCustomMapInput)

RequestContext(clientContext.userId, clientContext.guestId, featureContext)

}

private[configapi] def getFeatureContext(

clientContext: ClientContext,

product: Product,

featureOverrides: Map[String, FeatureValue],

fsCustomMapInput: Map[String, Any]

): FeatureContext = {

val recipient = getFeatureSwitchRecipient(clientContext)

.withCustomFields("product" -> product.identifier.toString, fsCustomMapInput.toSeq: \_\*)

val results = featureSwitches.matchRecipient(recipient)

OrElseFeatureContext(

ForcedFeatureContext(featureOverrides),

new FeatureSwitchResultsFeatureContext(results))

}

private[configapi] def getFeatureSwitchRecipient(

clientContext: ClientContext

): FeatureSwitchRecipient = FeatureSwitchRecipient(

userId = clientContext.userId,

userRoles = clientContext.userRoles,

deviceId = clientContext.deviceId,

guestId = clientContext.guestId,

languageCode = clientContext.languageCode,

countryCode = clientContext.countryCode,

userAgent = clientContext.userAgent.flatMap(UserAgent.apply),

isVerified = None,

clientApplicationId = clientContext.appId,

isTwoffice = clientContext.isTwoffice

)

}