package com.twitter.follow\_recommendations.modules

import com.google.inject.Provides

import com.twitter.abdecider.LoggingABDecider

import com.twitter.featureswitches.v2.Feature

import com.twitter.featureswitches.v2.FeatureFilter

import com.twitter.featureswitches.v2.FeatureSwitches

import com.twitter.featureswitches.v2.builder.FeatureSwitchesBuilder

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.follow\_recommendations.common.constants.GuiceNamedConstants.PRODUCER\_SIDE\_FEATURE\_SWITCHES

import com.twitter.inject.TwitterModule

import javax.inject.Named

import javax.inject.Singleton

object FeaturesSwitchesModule extends TwitterModule {

private val DefaultConfigRepoPath = "/usr/local/config"

private val FeaturesPath = "/features/onboarding/follow-recommendations-service/main"

val isLocal = flag("configrepo.local", false, "Is the server running locally or in a DC")

val localConfigRepoPath = flag(

"local.configrepo",

System.getProperty("user.home") + "/workspace/config",

"Path to your local config repo"

)

@Provides

@Singleton

def providesFeatureSwitches(

abDecider: LoggingABDecider,

statsReceiver: StatsReceiver

): FeatureSwitches = {

val configRepoPath = if (isLocal()) {

localConfigRepoPath()

} else {

DefaultConfigRepoPath

}

FeatureSwitchesBuilder

.createDefault(FeaturesPath, abDecider, Some(statsReceiver))

.configRepoAbsPath(configRepoPath)

.serviceDetailsFromAurora()

.build()

}

@Provides

@Singleton

@Named(PRODUCER\_SIDE\_FEATURE\_SWITCHES)

def providesProducerFeatureSwitches(

abDecider: LoggingABDecider,

statsReceiver: StatsReceiver

): FeatureSwitches = {

val configRepoPath = if (isLocal()) {

localConfigRepoPath()

} else {

DefaultConfigRepoPath

}

/\*\*

\* Feature Switches evaluate all tied FS Keys on Params construction time, which is very inefficient

\* for producer/candidate side holdbacks because we have 100s of candidates, and 100s of FS which result

\* in 10,000 FS evaluations when we want 1 per candidate (100 total), so we create a new FS Client

\* which has a [[ProducerFeatureFilter]] set for feature filter to reduce the FS Keys we evaluate.

\*/

FeatureSwitchesBuilder

.createDefault(FeaturesPath, abDecider, Some(statsReceiver.scope("producer\_side\_fs")))

.configRepoAbsPath(configRepoPath)

.serviceDetailsFromAurora()

.addFeatureFilter(ProducerFeatureFilter)

.build()

}

}

case object ProducerFeatureFilter extends FeatureFilter {

private val AllowedKeys = Set(

"post\_nux\_ml\_flow\_candidate\_user\_scorer\_id",

"frs\_receiver\_holdback\_keep\_social\_user\_candidate",

"frs\_receiver\_holdback\_keep\_user\_candidate")

override def filter(feature: Feature): Option[Feature] = {

if (AllowedKeys.exists(feature.parameters.contains)) {

Some(feature)

} else {

None

}

}

}