package com.twitter.follow\_recommendations.modules

import com.google.inject.Provides

import com.google.inject.Singleton

import com.google.inject.name.Named

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.follow\_recommendations.common.constants.GuiceNamedConstants

import com.twitter.inject.TwitterModule

import com.twitter.logging.BareFormatter

import com.twitter.logging.HandlerFactory

import com.twitter.logging.Level

import com.twitter.logging.LoggerFactory

import com.twitter.logging.NullHandler

import com.twitter.logging.QueueingHandler

import com.twitter.logging.ScribeHandler

object ScribeModule extends TwitterModule {

val useProdLogger = flag(

name = "scribe.use\_prod\_loggers",

default = false,

help = "whether to use production logging for service"

)

@Provides

@Singleton

@Named(GuiceNamedConstants.CLIENT\_EVENT\_LOGGER)

def provideClientEventsLoggerFactory(stats: StatsReceiver): LoggerFactory = {

val loggerCategory = "client\_event"

val clientEventsHandler: HandlerFactory = if (useProdLogger()) {

QueueingHandler(

maxQueueSize = 10000,

handler = ScribeHandler(

category = loggerCategory,

formatter = BareFormatter,

level = Some(Level.INFO),

statsReceiver = stats.scope("client\_event\_scribe")

)

)

} else { () => NullHandler }

LoggerFactory(

node = "abdecider",

level = Some(Level.INFO),

useParents = false,

handlers = clientEventsHandler :: Nil

)

}

@Provides

@Singleton

@Named(GuiceNamedConstants.REQUEST\_LOGGER)

def provideFollowRecommendationsLoggerFactory(stats: StatsReceiver): LoggerFactory = {

val loggerCategory = "follow\_recommendations\_logs"

val handlerFactory: HandlerFactory = if (useProdLogger()) {

QueueingHandler(

maxQueueSize = 10000,

handler = ScribeHandler(

category = loggerCategory,

formatter = BareFormatter,

level = Some(Level.INFO),

statsReceiver = stats.scope("follow\_recommendations\_logs\_scribe")

)

)

} else { () => NullHandler }

LoggerFactory(

node = loggerCategory,

level = Some(Level.INFO),

useParents = false,

handlers = handlerFactory :: Nil

)

}

@Provides

@Singleton

@Named(GuiceNamedConstants.FLOW\_LOGGER)

def provideFrsRecommendationFlowLoggerFactory(stats: StatsReceiver): LoggerFactory = {

val loggerCategory = "frs\_recommendation\_flow\_logs"

val handlerFactory: HandlerFactory = if (useProdLogger()) {

QueueingHandler(

maxQueueSize = 10000,

handler = ScribeHandler(

category = loggerCategory,

formatter = BareFormatter,

level = Some(Level.INFO),

statsReceiver = stats.scope("frs\_recommendation\_flow\_logs\_scribe")

)

)

} else { () => NullHandler }

LoggerFactory(

node = loggerCategory,

level = Some(Level.INFO),

useParents = false,

handlers = handlerFactory :: Nil

)

}

}