package com.twitter.cr\_mixer.module.core

import com.google.inject.Provides

import com.twitter.cr\_mixer.model.ModuleNames

import com.twitter.cr\_mixer.scribe.ScribeCategories

import com.twitter.cr\_mixer.scribe.ScribeCategory

import com.twitter.finagle.mtls.authentication.ServiceIdentifier

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.inject.TwitterModule

import com.twitter.logging.BareFormatter

import com.twitter.logging.Level

import com.twitter.logging.Logger

import com.twitter.logging.NullHandler

import com.twitter.logging.QueueingHandler

import com.twitter.logging.ScribeHandler

import com.twitter.logging.{LoggerFactory => TwitterLoggerFactory}

import javax.inject.Named

import javax.inject.Singleton

object LoggerFactoryModule extends TwitterModule {

private val DefaultQueueSize = 10000

@Provides

@Singleton

@Named(ModuleNames.AbDeciderLogger)

def provideAbDeciderLogger(

serviceIdentifier: ServiceIdentifier,

statsReceiver: StatsReceiver

): Logger = {

buildLoggerFactory(

ScribeCategories.AbDecider,

serviceIdentifier.environment,

statsReceiver.scope("ScribeLogger"))

.apply()

}

@Provides

@Singleton

@Named(ModuleNames.TopLevelApiDdgMetricsLogger)

def provideTopLevelApiDdgMetricsLogger(

serviceIdentifier: ServiceIdentifier,

statsReceiver: StatsReceiver

): Logger = {

buildLoggerFactory(

ScribeCategories.TopLevelApiDdgMetrics,

serviceIdentifier.environment,

statsReceiver.scope("ScribeLogger"))

.apply()

}

@Provides

@Singleton

@Named(ModuleNames.TweetRecsLogger)

def provideTweetRecsLogger(

serviceIdentifier: ServiceIdentifier,

statsReceiver: StatsReceiver

): Logger = {

buildLoggerFactory(

ScribeCategories.TweetsRecs,

serviceIdentifier.environment,

statsReceiver.scope("ScribeLogger"))

.apply()

}

@Provides

@Singleton

@Named(ModuleNames.BlueVerifiedTweetRecsLogger)

def provideVITTweetRecsLogger(

serviceIdentifier: ServiceIdentifier,

statsReceiver: StatsReceiver

): Logger = {

buildLoggerFactory(

ScribeCategories.VITTweetsRecs,

serviceIdentifier.environment,

statsReceiver.scope("ScribeLogger"))

.apply()

}

@Provides

@Singleton

@Named(ModuleNames.RelatedTweetsLogger)

def provideRelatedTweetsLogger(

serviceIdentifier: ServiceIdentifier,

statsReceiver: StatsReceiver

): Logger = {

buildLoggerFactory(

ScribeCategories.RelatedTweets,

serviceIdentifier.environment,

statsReceiver.scope("ScribeLogger"))

.apply()

}

@Provides

@Singleton

@Named(ModuleNames.UtegTweetsLogger)

def provideUtegTweetsLogger(

serviceIdentifier: ServiceIdentifier,

statsReceiver: StatsReceiver

): Logger = {

buildLoggerFactory(

ScribeCategories.UtegTweets,

serviceIdentifier.environment,

statsReceiver.scope("ScribeLogger"))

.apply()

}

@Provides

@Singleton

@Named(ModuleNames.AdsRecommendationsLogger)

def provideAdsRecommendationsLogger(

serviceIdentifier: ServiceIdentifier,

statsReceiver: StatsReceiver

): Logger = {

buildLoggerFactory(

ScribeCategories.AdsRecommendations,

serviceIdentifier.environment,

statsReceiver.scope("ScribeLogger"))

.apply()

}

private def buildLoggerFactory(

category: ScribeCategory,

environment: String,

statsReceiver: StatsReceiver

): TwitterLoggerFactory = {

environment match {

case "prod" =>

TwitterLoggerFactory(

node = category.getProdLoggerFactoryNode,

level = Some(Level.INFO),

useParents = false,

handlers = List(

QueueingHandler(

maxQueueSize = DefaultQueueSize,

handler = ScribeHandler(

category = category.scribeCategory,

formatter = BareFormatter,

statsReceiver = statsReceiver.scope(category.getProdLoggerFactoryNode)

)

)

)

)

case \_ =>

TwitterLoggerFactory(

node = category.getStagingLoggerFactoryNode,

level = Some(Level.DEBUG),

useParents = false,

handlers = List(

{ () => NullHandler }

)

)

}

}

}