package com.twitter.frigate.pushservice

import com.twitter.discovery.common.environment.modules.EnvironmentModule

import com.twitter.finagle.Filter

import com.twitter.finatra.annotations.DarkTrafficFilterType

import com.twitter.finatra.decider.modules.DeciderModule

import com.twitter.finatra.http.HttpServer

import com.twitter.finatra.http.filters.CommonFilters

import com.twitter.finatra.http.routing.HttpRouter

import com.twitter.finatra.mtls.http.{Mtls => HttpMtls}

import com.twitter.finatra.mtls.thriftmux.{Mtls => ThriftMtls}

import com.twitter.finatra.mtls.thriftmux.filters.MtlsServerSessionTrackerFilter

import com.twitter.finatra.thrift.ThriftServer

import com.twitter.finatra.thrift.filters.ExceptionMappingFilter

import com.twitter.finatra.thrift.filters.LoggingMDCFilter

import com.twitter.finatra.thrift.filters.StatsFilter

import com.twitter.finatra.thrift.filters.ThriftMDCFilter

import com.twitter.finatra.thrift.filters.TraceIdMDCFilter

import com.twitter.finatra.thrift.routing.ThriftRouter

import com.twitter.frigate.common.logger.MRLoggerGlobalVariables

import com.twitter.frigate.pushservice.controller.PushServiceController

import com.twitter.frigate.pushservice.module.\_

import com.twitter.inject.TwitterModule

import com.twitter.inject.annotations.Flags

import com.twitter.inject.thrift.modules.ThriftClientIdModule

import com.twitter.logging.BareFormatter

import com.twitter.logging.Level

import com.twitter.logging.LoggerFactory

import com.twitter.logging.{Logging => JLogging}

import com.twitter.logging.QueueingHandler

import com.twitter.logging.ScribeHandler

import com.twitter.product\_mixer.core.module.product\_mixer\_flags.ProductMixerFlagModule

import com.twitter.product\_mixer.core.module.ABDeciderModule

import com.twitter.product\_mixer.core.module.FeatureSwitchesModule

import com.twitter.product\_mixer.core.module.StratoClientModule

object PushServiceMain extends PushServiceFinatraServer

class PushServiceFinatraServer

extends ThriftServer

with ThriftMtls

with HttpServer

with HttpMtls

with JLogging {

override val name = "PushService"

override val modules: Seq[TwitterModule] = {

Seq(

ABDeciderModule,

DeciderModule,

FeatureSwitchesModule,

FilterModule,

FlagModule,

EnvironmentModule,

ThriftClientIdModule,

DeployConfigModule,

ProductMixerFlagModule,

StratoClientModule,

PushHandlerModule,

PushTargetUserBuilderModule,

PushServiceDarkTrafficModule,

LoggedOutPushTargetUserBuilderModule,

new ThriftWebFormsModule(this),

)

}

override def configureThrift(router: ThriftRouter): Unit = {

router

.filter[ExceptionMappingFilter]

.filter[LoggingMDCFilter]

.filter[TraceIdMDCFilter]

.filter[ThriftMDCFilter]

.filter[MtlsServerSessionTrackerFilter]

.filter[StatsFilter]

.filter[Filter.TypeAgnostic, DarkTrafficFilterType]

.add[PushServiceController]

}

override def configureHttp(router: HttpRouter): Unit =

router

.filter[CommonFilters]

override protected def start(): Unit = {

MRLoggerGlobalVariables.setRequiredFlags(

traceLogFlag = injector.instance[Boolean](Flags.named(FlagModule.mrLoggerIsTraceAll.name)),

nthLogFlag = injector.instance[Boolean](Flags.named(FlagModule.mrLoggerNthLog.name)),

nthLogValFlag = injector.instance[Long](Flags.named(FlagModule.mrLoggerNthVal.name))

)

}

override protected def warmup(): Unit = {

handle[PushMixerThriftServerWarmupHandler]()

}

override protected def configureLoggerFactories(): Unit = {

loggerFactories.foreach { \_() }

}

override def loggerFactories: List[LoggerFactory] = {

val scribeScope = statsReceiver.scope("scribe")

List(

LoggerFactory(

level = Some(levelFlag()),

handlers = handlers

),

LoggerFactory(

node = "request\_scribe",

level = Some(Level.INFO),

useParents = false,

handlers = QueueingHandler(

maxQueueSize = 10000,

handler = ScribeHandler(

category = "frigate\_pushservice\_log",

formatter = BareFormatter,

statsReceiver = scribeScope.scope("frigate\_pushservice\_log")

)

) :: Nil

),

LoggerFactory(

node = "notification\_scribe",

level = Some(Level.INFO),

useParents = false,

handlers = QueueingHandler(

maxQueueSize = 10000,

handler = ScribeHandler(

category = "frigate\_notifier",

formatter = BareFormatter,

statsReceiver = scribeScope.scope("frigate\_notifier")

)

) :: Nil

),

LoggerFactory(

node = "push\_scribe",

level = Some(Level.INFO),

useParents = false,

handlers = QueueingHandler(

maxQueueSize = 10000,

handler = ScribeHandler(

category = "test\_frigate\_push",

formatter = BareFormatter,

statsReceiver = scribeScope.scope("test\_frigate\_push")

)

) :: Nil

),

LoggerFactory(

node = "push\_subsample\_scribe",

level = Some(Level.INFO),

useParents = false,

handlers = QueueingHandler(

maxQueueSize = 2500,

handler = ScribeHandler(

category = "magicrecs\_candidates\_subsample\_scribe",

maxMessagesPerTransaction = 250,

maxMessagesToBuffer = 2500,

formatter = BareFormatter,

statsReceiver = scribeScope.scope("magicrecs\_candidates\_subsample\_scribe")

)

) :: Nil

),

LoggerFactory(

node = "mr\_request\_scribe",

level = Some(Level.INFO),

useParents = false,

handlers = QueueingHandler(

maxQueueSize = 2500,

handler = ScribeHandler(

category = "mr\_request\_scribe",

maxMessagesPerTransaction = 250,

maxMessagesToBuffer = 2500,

formatter = BareFormatter,

statsReceiver = scribeScope.scope("mr\_request\_scribe")

)

) :: Nil

),

LoggerFactory(

node = "high\_quality\_candidates\_scribe",

level = Some(Level.INFO),

useParents = false,

handlers = QueueingHandler(

maxQueueSize = 2500,

handler = ScribeHandler(

category = "frigate\_high\_quality\_candidates\_log",

maxMessagesPerTransaction = 250,

maxMessagesToBuffer = 2500,

formatter = BareFormatter,

statsReceiver = scribeScope.scope("high\_quality\_candidates\_scribe")

)

) :: Nil

),

)

}

}