package com.twitter.frigate.pushservice.module

import com.google.inject.Provides

import com.google.inject.Singleton

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.frigate.pushservice.target.LoggedOutPushTargetUserBuilder

import com.twitter.frigate.pushservice.refresh\_handler.RefreshForPushHandler

import com.twitter.frigate.pushservice.config.DeployConfig

import com.twitter.frigate.pushservice.send\_handler.SendHandler

import com.twitter.frigate.pushservice.take.candidate\_validator.RFPHCandidateValidator

import com.twitter.frigate.pushservice.take.candidate\_validator.SendHandlerPostCandidateValidator

import com.twitter.frigate.pushservice.take.candidate\_validator.SendHandlerPreCandidateValidator

import com.twitter.frigate.pushservice.refresh\_handler.LoggedOutRefreshForPushHandler

import com.twitter.frigate.pushservice.take.SendHandlerNotifier

import com.twitter.frigate.pushservice.target.PushTargetUserBuilder

import com.twitter.inject.TwitterModule

object PushHandlerModule extends TwitterModule {

@Provides

@Singleton

def providesRefreshForPushHandler(

pushTargetUserBuilder: PushTargetUserBuilder,

config: DeployConfig,

statsReceiver: StatsReceiver

): RefreshForPushHandler = {

new RefreshForPushHandler(

pushTargetUserBuilder = pushTargetUserBuilder,

candSourceGenerator = config.candidateSourceGenerator,

rfphRanker = config.rfphRanker,

candidateHydrator = config.candidateHydrator,

candidateValidator = new RFPHCandidateValidator(config),

rfphTakeStepUtil = config.rfphTakeStepUtil,

rfphRestrictStep = config.rfphRestrictStep,

rfphNotifier = config.rfphNotifier,

rfphStatsRecorder = config.rfphStatsRecorder,

mrRequestScriberNode = config.mrRequestScriberNode,

rfphFeatureHydrator = config.rfphFeatureHydrator,

rfphPrerankFilter = config.rfphPrerankFilter,

rfphLightRanker = config.rfphLightRanker

)(statsReceiver)

}

@Provides

@Singleton

def providesSendHandler(

pushTargetUserBuilder: PushTargetUserBuilder,

config: DeployConfig,

statsReceiver: StatsReceiver

): SendHandler = {

new SendHandler(

pushTargetUserBuilder,

new SendHandlerPreCandidateValidator(config),

new SendHandlerPostCandidateValidator(config),

new SendHandlerNotifier(config.candidateNotifier, statsReceiver.scope("SendHandlerNotifier")),

config.sendHandlerCandidateHydrator,

config.featureHydrator,

config.sendHandlerPredicateUtil,

config.mrRequestScriberNode)(statsReceiver, config)

}

@Provides

@Singleton

def providesLoggedOutRefreshForPushHandler(

loPushTargetUserBuilder: LoggedOutPushTargetUserBuilder,

config: DeployConfig,

statsReceiver: StatsReceiver

): LoggedOutRefreshForPushHandler = {

new LoggedOutRefreshForPushHandler(

loPushTargetUserBuilder = loPushTargetUserBuilder,

loPushCandidateSourceGenerator = config.loCandidateSourceGenerator,

candidateHydrator = config.candidateHydrator,

loRanker = config.loggedOutRFPHRanker,

loRfphNotifier = config.loRfphNotifier,

loMrRequestScriberNode = config.loggedOutMrRequestScriberNode,

)(statsReceiver)

}

}