Externalized properties

* property-source
* property-source-env
* property-source-multifile
* property-source-spring-boot

Profile specific

* spring-profiles
* spring-profiles-unit-test
* spring-profiles-runtime
* profile-properties
* properites-code-assign
* profile-yaml-props

mysql

* mysql-maven
* mysql-qa-datasource
* service-account
* code-assignment-complete

CI

* [https://tecadmin.net/install-java-8-on-centos-rhel-and-fedora/#](https://tecadmin.net/install-java-8-on-centos-rhel-and-fedora/)
* <https://wiki.jenkins.io/display/JENKINS/Installing+Jenkins+on+Red+Hat+distributions>

artifactory

* <https://docs.docker.com/install/linux/docker-ee/rhel/>
* https://www.jfrog.com/confluence/display/RTF/Configuring+Apache
* artifactory-distro-config
* artifactory-docker
* artifactory-resolving

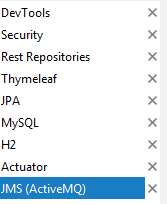
virtual cloud deployment

* prod-mysql-config
* running-spring-boot-application
* springbootservice

rds

* rds-profile

start project



Exercise 1（specify the properties file name）

external properties @Value @propertysource @configure @bean

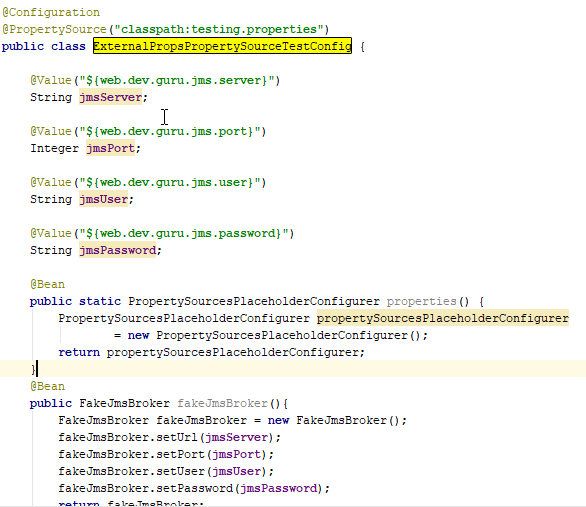
load the specified testing.properties file into the bean class fakejmsbroker, in the config class externalpropsproperty source config, assembly the bean using the @value

bean fakejmsbroker



Config class using @propertysource@configure @bean

Need to specify the propertysourceplaceholderconfigure class



Test the config file using @contextconfiguration

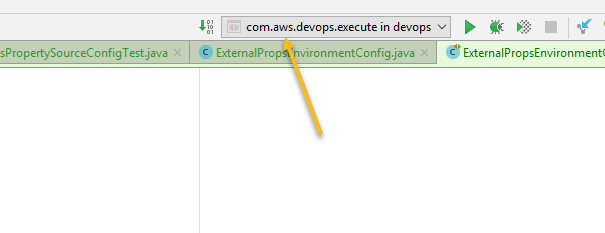


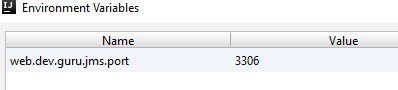
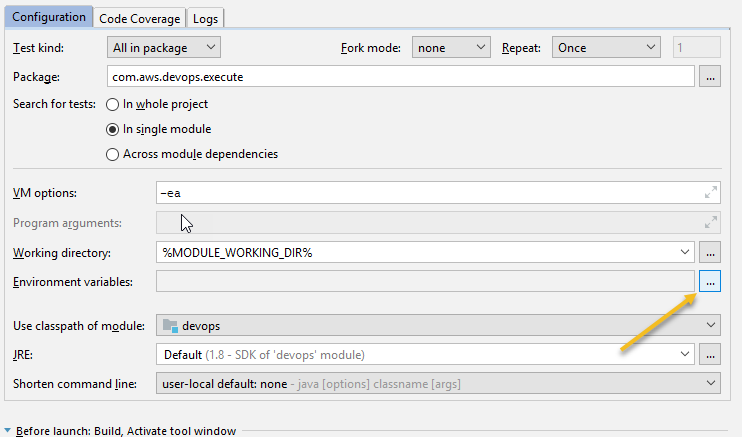
Exercise 2 (using Emviroment class)

Instead of using @value and placeholder bean,using environment class

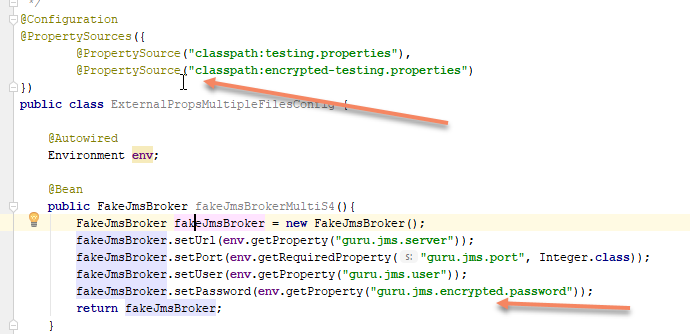


Change Environment variable



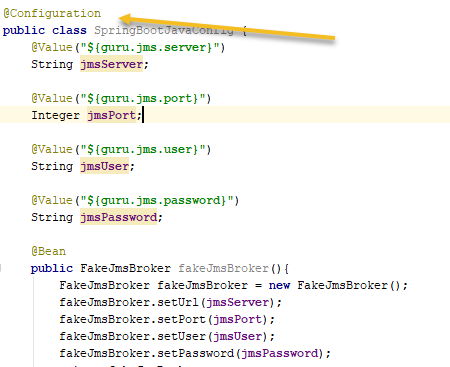


Exercise 3 (Multiple property file using using propertysources(propertysource(classpath:))



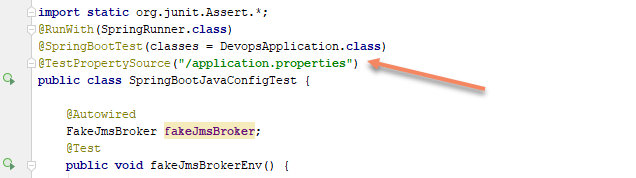
Exercise 4 (Springboot integration test)

Set the properties file in the runtime instead of in the config file



There is no propertysource under the @configuration

Need to specify the endpoint of real class and specify the real property file

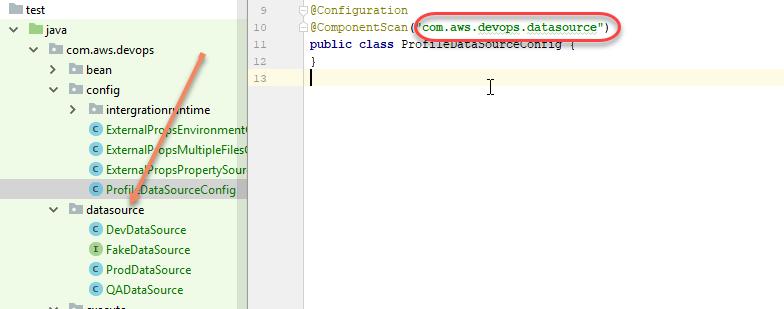


Exercise 5 (Profile @activeprofiles)

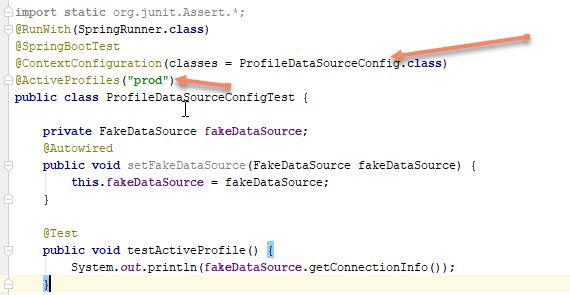
Define multiple class, each class mapping one profile



In the config file scan the class folder path



In the test file, specify the config file and active profile value.



* Set default profile

@profile(“dev”,”default”)

@activeprofile(“dev”) for unittest only

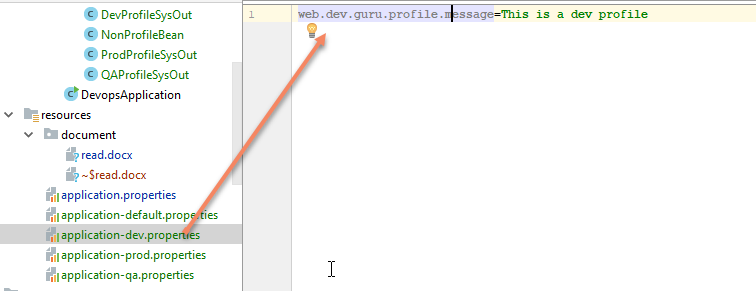
**spring.profiles.active**=**dev**

Hierarchy

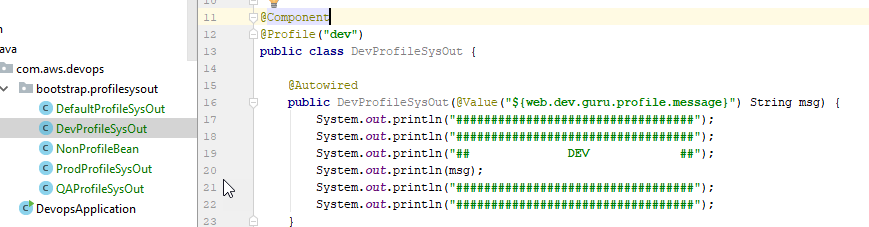
Profile value in properties file is higher than default

Exercise 6 (profile properties)

Create multiple properties files when profile value in the middle of file name



Each class will mapping one properties



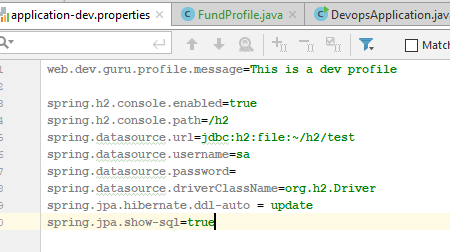
When run spring boot, set spring.profile.active in the application.properties file

higher than

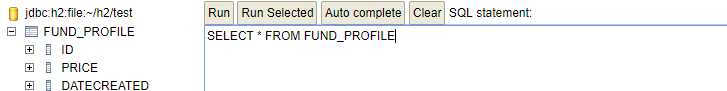
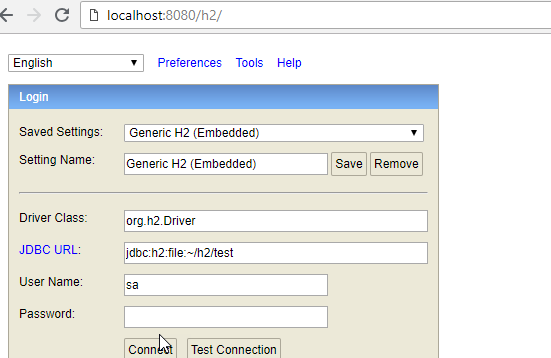
Properties file value.

H2

in the application-dev.properties file

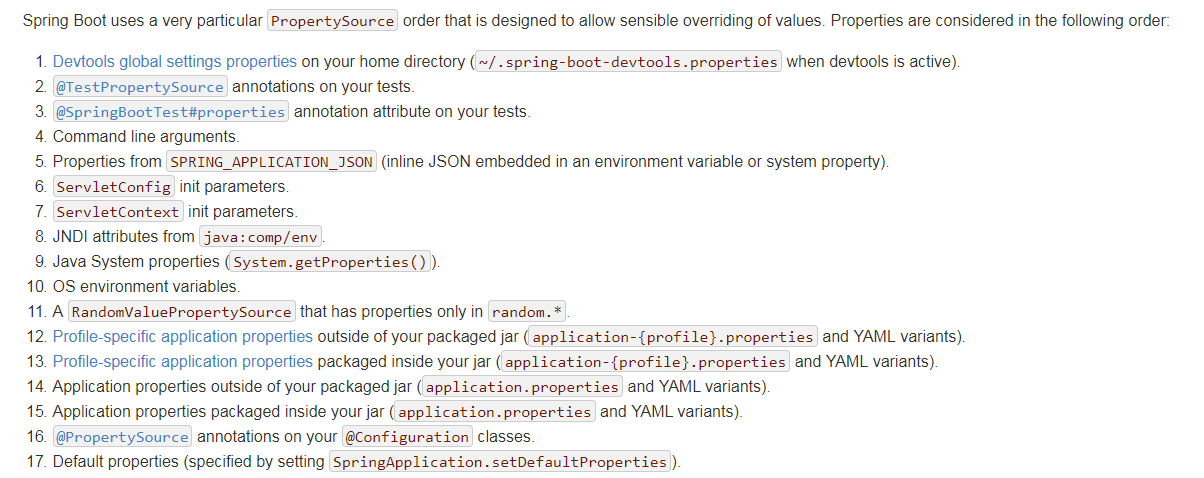


Start the server and access the url



Properties load hierarchy

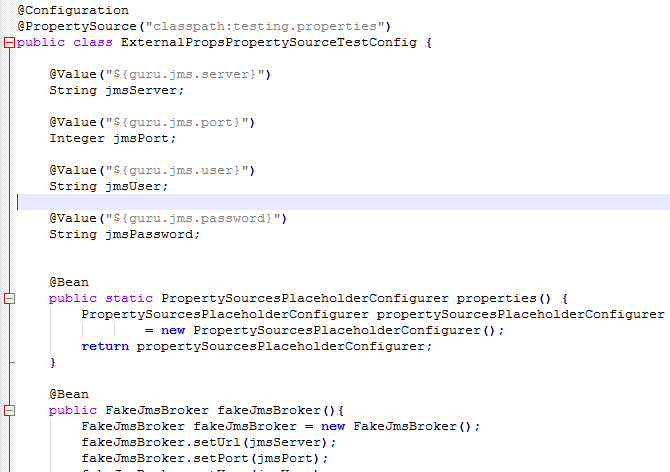
https://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-external-config.html



Load external properties when integration test

runtime

@value properties



Environment property



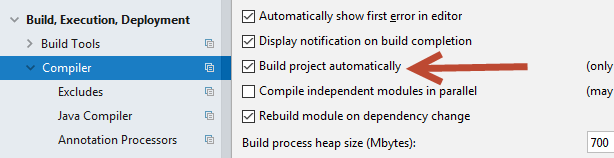
@profile

When specify the profile value, the class with the profile will be active

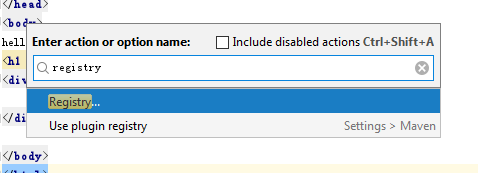
1. Define three datasource classes.(devdatasource,qadatasource,proddatasource)
2. @profile(“dev”) @profile(“qa”) @profile(“prod”) in front of each datasource
3. @componentscan(“ all datasource class path”)

Hotswap solution

1. Devops dependency



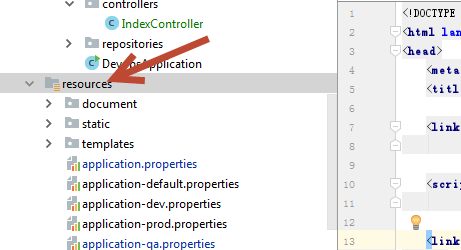
1. Ctrl+shift+A



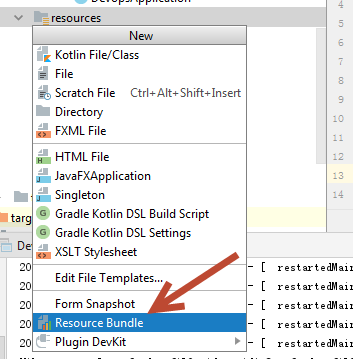
compiler.automake.allow.when.app.running

international message bundle

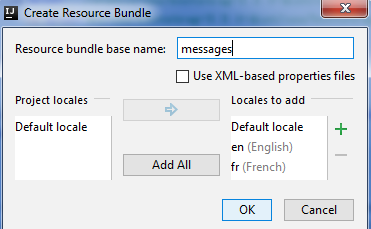
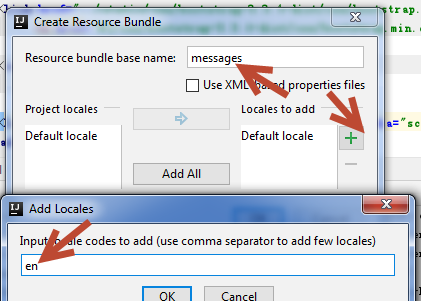
* cursor on the resources folder

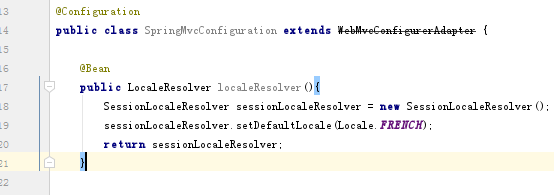
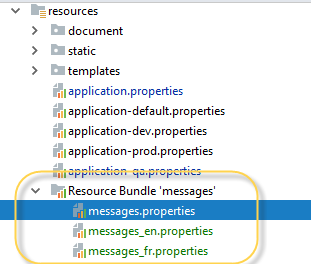


* Alt+Insert



Add messages and add en/fr





RDS vs DB ec2 instance

There’s almost no manual configuration, management, or maintenance requirements – AWS even handles software patches automatically.

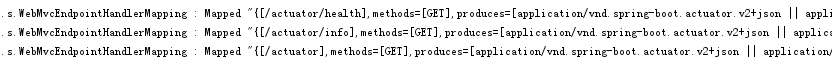
Each RDS instance is pre-configured and optimized based on the selected size. RDS instances scale by adjusting memory or compute power up or down as needed.

RDS also features the intangible 15% performance boost over an EC2 MySQL instance as per our internal benchmarking and also as seen in this article. The reason for this performance boost is because RDS is running on better I/O hardware (and with RAID as per our assumption) than a vanilla EC2 instance backed by EBS.

It’s also simple to configure read replicas or set up synchronous replication across availability zones for enhanced performance, availability, and durability.

RDS also comes with automated scheduled backups and point in time restores

Spring actuator



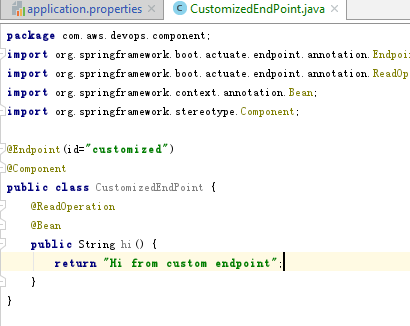
By default only 2 endpoint ,add this in application.properties for all endpoint

**management.endpoints.web.exposure.include**=**\***

**others**

**management.endpoints.web.exposure.exclude= # Endpoint IDs that should be excluded.**

Create customized endpoint



Register the endpoint into application.properties

**management.endpoints.web.exposure.include**=**customized,info,health…**

encrypt the password in properties file

<**dependency**>  
 <**groupId**>com.github.ulisesbocchio</**groupId**>  
 <**artifactId**>jasypt-spring-boot-starter</**artifactId**>  
 <**version**>2.0.0</**version**>  
</**dependency**>

Set salt password

C:\Development\workspaceJava\web-dev-guru\spring-boot-devops-inhouse\devops\utils\jasypt-1.9.2\bin>encrypt.bat input=root password=password

Username =root

Password =Hadoop

Salt=’password’

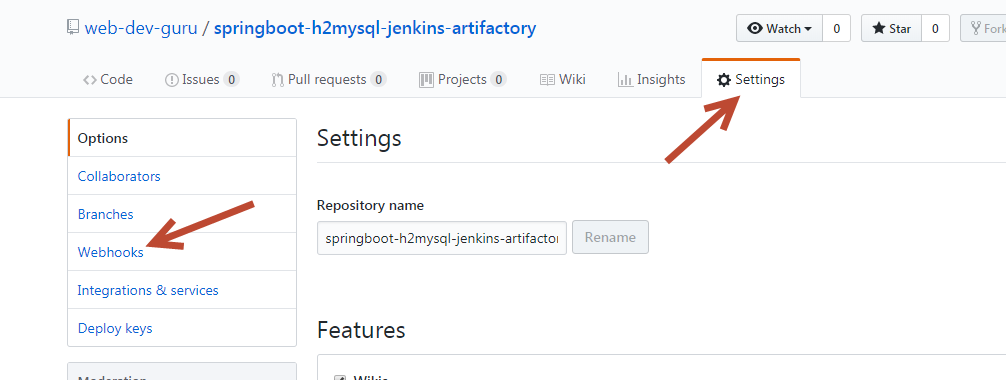
Generate two encrypt value for username and password

Copy it to application-qa.properties

**jasypt.encryptor.password**=**password**

**spring.datasource.url**=**jdbc:mysql://localhost:3306/world***#spring.datasource.username=root  
#spring.datasource.password=hadoop***spring.datasource.username**=**ENC(iM/RWLJl3RTrDVuG/WliVw==)  
spring.datasource.password**=**ENC(wy2JvdAfFNaepDPIp+frFA==)**

add webhooker in the github



Github-webhook is required

