**SpringBoot**

**1.SpringBoot介绍:**

随着动态语言的流行(Ruby,Groovy,Scala,Node.js),Java的开发显得格外的笨重：繁多的配置，低下的开发效率，复杂的部署流程以及第三方技术集成难度大。

在上述环境下，SpringBoot应运而生。它使用“习惯由于配置”的理念让项目快速运行起来。使用SpringBoot很容易创建一个独立运行(运行jar，内嵌Servlet容器)准生产级别的基于Spring框架的项目，使用SpringBoot可以不用或只需要很少的Spring配置。

1. **SpringBoot核心特点**：
2. 可以以jar包的形式独立运行，运行一个SpringBoot项目只需通过 java -jar xx.jar来运行
3. 内嵌Servlet容器，SpringBoot可以选择Tomcat，Jetty或者Undertow，这样我们无须以war包形式部署项目
4. 简化Maven配置，SpringBoot提供了一系列的starter pom 来简化Maven的依赖加载
5. SpringBoot会根据在类路径中的jar包，类，为jar包中的类自动配置Bean，这样就极大的减少了我们要使用的配置
6. SpringBoot提供了基于http，ssh，telnet对运行时的项目进行监控
7. 不借助于代码生成来实现，而是通过条件注解来实现，这也是Spring4.x的新特性，不需要任何的xml配置即可实现Spring的所有配置
8. **搭建第一个SpringBoot入门程序：**

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>1.5.6.RELEASE</version>

</parent>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

改变JDK版本:

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

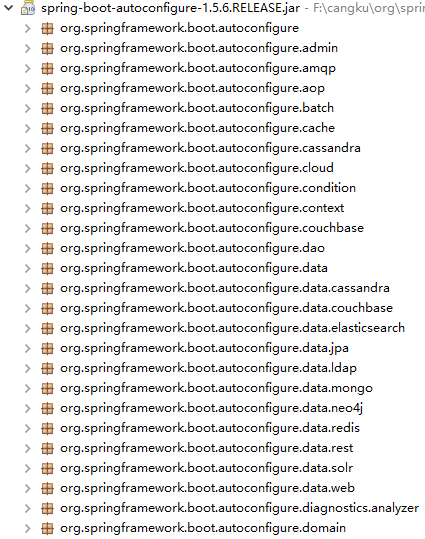
</plugin>

</plugins>

</build>

1. @EnableAutoConfiguration:启用自动配置，该注解会使SpringBoot根据项目依赖的jar包自动配置项目的配置项。例如：我们添加了spring-boot-starter-web的依赖，项目中也就会引入SpringMVC的依赖，Spring Boot就会自动配置Tomcat和SpringMVC。

Spring Boot支持的自动配置如下：



1. 关闭自动配置：

@EnableAutoConfiguration(exclude={RedisAutoConfiguration.**class**})

**6.**自定义banner：

1. 打开网站：

[http://patorjk.com/software/taag/#p=display&h=3&v=3&f=4Max&t=qf%20Spring%20Boot](http://patorjk.com/software/taag/#p=display&h=3&v=3&f=4Max&t=itcast%20Spring%20Boot)

1. 拷贝生成的字符到一个文本文件中，并且将该文件命名为banner.txt

将banner.txt拷贝到项目的resources目录中

1. 将banner.txt拷贝到项目的resources目录中：

### 7.全局配置文件

Spring Boot项目使用一个全局的配置文件application.properties或者是application.yml，在resources目录下或者类路径下的/config下，一般我们放到resources下。

修改tomcat的端口为8088：

server.port=8088

进入DispatcherServlet的规则为：\*.html  
server.context-path=/hello

*# BANNER*

banner.charset=UTF-8 *# Banner file encoding.*

banner.location=classpath:banner.txt *# Banner file location.*

banner.image.location=classpath:banner.gif *# Banner image file location (jpg/png can also be used).*

banner.image.width= *# Width of the banner image in chars (default 76)*

banner.image.height= *# Height of the banner image in chars (default based on image height)*

banner.image.margin= *# Left hand image margin in chars (default 2)*

banner.image.invert= *# If images should be inverted for dark terminal themes (default false)*

*# LOGGING*

logging.config= *# Location of the logging configuration file. For instance `classpath:logback.xml` for Logback*

logging.exception-conversion-word=%wEx *# Conversion word used when logging exceptions.*

logging.file= *# Log file name. For instance `myapp.log`*

logging.level.\*= *# Log levels severity mapping. For instance `logging.level.org.springframework=DEBUG`*

logging.path= *# Location of the log file. For instance `/var/log`*

logging.pattern.console= *# Appender pattern for output to the console. Only supported with the default logback setup.*

logging.pattern.file= *# Appender pattern for output to the file. Only supported with the default logback setup.*

logging.pattern.level= *# Appender pattern for log level (default %5p). Only supported with the default logback setup.*

logging.register-shutdown-hook=false *# Register a shutdown hook for the logging system when it is initialized.*

*# AOP*

spring.aop.auto=true *# Add @EnableAspectJAutoProxy.*

spring.aop.proxy-target-class=false *# Whether subclass-based (CGLIB) proxies are to be created (true) as opposed to standard Java interface-based proxies (false).*

*# IDENTITY (*[ContextIdApplicationContextInitializer](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot/src/main/java/org/springframework/boot/context/ContextIdApplicationContextInitializer.java))

spring.application.index= *# Application index.*

spring.application.name= *# Application name.*

*# ADMIN (*[SpringApplicationAdminJmxAutoConfiguration](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/admin/SpringApplicationAdminJmxAutoConfiguration.java))

spring.application.admin.enabled=false *# Enable admin features for the application.*

spring.application.admin.jmx-name=org.springframework.boot:type=Admin,name=SpringApplication *# JMX name of the application admin MBean.*

*# AUTO-CONFIGURATION*

spring.autoconfigure.exclude= *# Auto-configuration classes to exclude.*

*# SPRING CORE*

spring.beaninfo.ignore=true *# Skip search of BeanInfo classes.*

*# SPRING CACHE (*[CacheProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/cache/CacheProperties.java))

spring.cache.cache-names= *# Comma-separated list of cache names to create if supported by the underlying cache manager.*

spring.cache.caffeine.spec= *# The spec to use to create caches. Check CaffeineSpec for more details on the spec format.*

spring.cache.couchbase.expiration=0 *# Entry expiration in milliseconds. By default the entries never expire.*

spring.cache.ehcache.config= *# The location of the configuration file to use to initialize EhCache.*

spring.cache.guava.spec= *# The spec to use to create caches. Check CacheBuilderSpec for more details on the spec format.*

spring.cache.infinispan.config= *# The location of the configuration file to use to initialize Infinispan.*

spring.cache.jcache.config= *# The location of the configuration file to use to initialize the cache manager.*

spring.cache.jcache.provider= *# Fully qualified name of the CachingProvider implementation to use to retrieve the JSR-107 compliant cache manager. Only needed if more than one JSR-107 implementation is available on the classpath.*

spring.cache.type= *# Cache type, auto-detected according to the environment by default.*

*# SPRING CONFIG - using environment property only (*[ConfigFileApplicationListener](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot/src/main/java/org/springframework/boot/context/config/ConfigFileApplicationListener.java))

spring.config.location= *# Config file locations.*

spring.config.name=application *# Config file name.*

*# HAZELCAST (*[HazelcastProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/hazelcast/HazelcastProperties.java))

spring.hazelcast.config= *# The location of the configuration file to use to initialize Hazelcast.*

*# PROJECT INFORMATION (*[ProjectInfoProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/info/ProjectInfoProperties.java))

spring.info.build.location=classpath:META-INF/build-info.properties *# Location of the generated build-info.properties file.*

spring.info.git.location=classpath:git.properties *# Location of the generated git.properties file.*

*# JMX*

spring.jmx.default-domain= *# JMX domain name.*

spring.jmx.enabled=true *# Expose management beans to the JMX domain.*

spring.jmx.server=mbeanServer *# MBeanServer bean name.*

*# Email (*[MailProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/mail/MailProperties.java))

spring.mail.default-encoding=UTF-8 *# Default MimeMessage encoding.*

spring.mail.host= *# SMTP server host. For instance `smtp.example.com`*

spring.mail.jndi-name= *# Session JNDI name. When set, takes precedence to others mail settings.*

spring.mail.password= *# Login password of the SMTP server.*

spring.mail.port= *# SMTP server port.*

spring.mail.properties.\*= *# Additional JavaMail session properties.*

spring.mail.protocol=smtp *# Protocol used by the SMTP server.*

spring.mail.test-connection=false *# Test that the mail server is available on startup.*

spring.mail.username= *# Login user of the SMTP server.*

*# APPLICATION SETTINGS (*[SpringApplication](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot/src/main/java/org/springframework/boot/SpringApplication.java))

spring.main.banner-mode=console *# Mode used to display the banner when the application runs.*

spring.main.sources= *# Sources (class name, package name or XML resource location) to include in the ApplicationContext.*

spring.main.web-environment= *# Run the application in a web environment (auto-detected by default).*

*# FILE ENCODING (*[FileEncodingApplicationListener](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot/src/main/java/org/springframework/boot/context/FileEncodingApplicationListener.java))

spring.mandatory-file-encoding= *# Expected character encoding the application must use.*

*# INTERNATIONALIZATION (*[MessageSourceAutoConfiguration](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/context/MessageSourceAutoConfiguration.java))

spring.messages.always-use-message-format=false *# Set whether to always apply the MessageFormat rules, parsing even messages without arguments.*

spring.messages.basename=messages *# Comma-separated list of basenames, each following the ResourceBundle convention.*

spring.messages.cache-seconds=-1 *# Loaded resource bundle files cache expiration, in seconds. When set to -1, bundles are cached forever.*

spring.messages.encoding=UTF-8 *# Message bundles encoding.*

spring.messages.fallback-to-system-locale=true *# Set whether to fall back to the system Locale if no files for a specific Locale have been found.*

*# OUTPUT*

spring.output.ansi.enabled=detect *# Configure the ANSI output.*

*# PID FILE (*[ApplicationPidFileWriter](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot/src/main/java/org/springframework/boot/system/ApplicationPidFileWriter.java))

spring.pid.fail-on-write-error= *# Fail if ApplicationPidFileWriter is used but it cannot write the PID file.*

spring.pid.file= *# Location of the PID file to write (if ApplicationPidFileWriter is used).*

*# PROFILES*

spring.profiles.active= *# Comma-separated list (or list if using YAML) of* [active profiles](http://docs.spring.io/spring-boot/docs/1.5.2.RELEASE/reference/htmlsingle/#howto-set-active-spring-profiles).

spring.profiles.include= *# Unconditionally activate the specified comma separated profiles (or list of profiles if using YAML).*

*# SENDGRID (*[SendGridAutoConfiguration](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/sendgrid/SendGridAutoConfiguration.java))

spring.sendgrid.api-key= *# SendGrid api key (alternative to username/password)*

spring.sendgrid.username= *# SendGrid account username*

spring.sendgrid.password= *# SendGrid account password*

spring.sendgrid.proxy.host= *# SendGrid proxy host*

spring.sendgrid.proxy.port= *# SendGrid proxy port*

*# ----------------------------------------*

*# WEB PROPERTIES*

*# ----------------------------------------*

*# EMBEDDED SERVER CONFIGURATION (*[ServerProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/web/ServerProperties.java))

server.address= *# Network address to which the server should bind to.*

server.compression.enabled=false *# If response compression is enabled.*

server.compression.excluded-user-agents= *# List of user-agents to exclude from compression.*

server.compression.mime-types= *# Comma-separated list of MIME types that should be compressed. For instance `text/html,text/css,application/json`*

server.compression.min-response-size= *# Minimum response size that is required for compression to be performed. For instance 2048*

server.connection-timeout= *# Time in milliseconds that connectors will wait for another HTTP request before closing the connection. When not set, the connector's container-specific default will be used. Use a value of -1 to indicate no (i.e. infinite) timeout.*

server.context-parameters.\*= *# Servlet context init parameters. For instance `server.context-parameters.a=alpha`*

server.context-path= *# Context path of the application.*

server.display-name=application *# Display name of the application.*

server.max-http-header-size=0 *# Maximum size in bytes of the HTTP message header.*

server.error.include-stacktrace=never *# When to include a "stacktrace" attribute.*

server.error.path=/error *# Path of the error controller.*

server.error.whitelabel.enabled=true *# Enable the default error page displayed in browsers in case of a server error.*

server.jetty.acceptors= *# Number of acceptor threads to use.*

server.jetty.max-http-post-size=0 *# Maximum size in bytes of the HTTP post or put content.*

server.jetty.selectors= *# Number of selector threads to use.*

server.jsp-servlet.class-name=org.apache.jasper.servlet.JspServlet *# The class name of the JSP servlet.*

server.jsp-servlet.init-parameters.\*= *# Init parameters used to configure the JSP servlet*

server.jsp-servlet.registered=true *# Whether or not the JSP servlet is registered*

server.port=8080 *# Server HTTP port.*

server.server-header= *# Value to use for the Server response header (no header is sent if empty)*

server.servlet-path=/ *# Path of the main dispatcher servlet.*

server.use-forward-headers= *# If X-Forwarded-\* headers should be applied to the HttpRequest.*

server.session.cookie.comment= *# Comment for the session cookie.*

server.session.cookie.domain= *# Domain for the session cookie.*

server.session.cookie.http-only= *# "HttpOnly" flag for the session cookie.*

server.session.cookie.max-age= *# Maximum age of the session cookie in seconds.*

server.session.cookie.name= *# Session cookie name.*

server.session.cookie.path= *# Path of the session cookie.*

server.session.cookie.secure= *# "Secure" flag for the session cookie.*

server.session.persistent=false *# Persist session data between restarts.*

server.session.store-dir= *# Directory used to store session data.*

server.session.timeout= *# Session timeout in seconds.*

server.session.tracking-modes= *# Session tracking modes (one or more of the following: "cookie", "url", "ssl").*

server.ssl.ciphers= *# Supported SSL ciphers.*

server.ssl.client-auth= *# Whether client authentication is wanted ("want") or needed ("need"). Requires a trust store.*

server.ssl.enabled= *# Enable SSL support.*

server.ssl.enabled-protocols= *# Enabled SSL protocols.*

server.ssl.key-alias= *# Alias that identifies the key in the key store.*

server.ssl.key-password= *# Password used to access the key in the key store.*

server.ssl.key-store= *# Path to the key store that holds the SSL certificate (typically a jks file).*

server.ssl.key-store-password= *# Password used to access the key store.*

server.ssl.key-store-provider= *# Provider for the key store.*

server.ssl.key-store-type= *# Type of the key store.*

server.ssl.protocol=TLS *# SSL protocol to use.*

server.ssl.trust-store= *# Trust store that holds SSL certificates.*

server.ssl.trust-store-password= *# Password used to access the trust store.*

server.ssl.trust-store-provider= *# Provider for the trust store.*

server.ssl.trust-store-type= *# Type of the trust store.*

server.tomcat.accept-count= *# Maximum queue length for incoming connection requests when all possible request processing threads are in use.*

server.tomcat.accesslog.buffered=true *# Buffer output such that it is only flushed periodically.*

server.tomcat.accesslog.directory=logs *# Directory in which log files are created. Can be relative to the tomcat base dir or absolute.*

server.tomcat.accesslog.enabled=false *# Enable access log.*

server.tomcat.accesslog.pattern=common *# Format pattern for access logs.*

server.tomcat.accesslog.prefix=access\_log *# Log file name prefix.*

server.tomcat.accesslog.rename-on-rotate=false *# Defer inclusion of the date stamp in the file name until rotate time.*

server.tomcat.accesslog.request-attributes-enabled=false *# Set request attributes for IP address, Hostname, protocol and port used for the request.*

server.tomcat.accesslog.rotate=true *# Enable access log rotation.*

server.tomcat.accesslog.suffix=.log *# Log file name suffix.*

server.tomcat.additional-tld-skip-patterns= *# Comma-separated list of additional patterns that match jars to ignore for TLD scanning.*

server.tomcat.background-processor-delay=30 *# Delay in seconds between the invocation of backgroundProcess methods.*

server.tomcat.basedir= *# Tomcat base directory. If not specified a temporary directory will be used.*

server.tomcat.internal-proxies=10\\.\\d{1,3}\\.\\d{1,3}\\.\\d{1,3}|\\

192\\.168\\.\\d{1,3}\\.\\d{1,3}|\\

169\\.254\\.\\d{1,3}\\.\\d{1,3}|\\

127\\.\\d{1,3}\\.\\d{1,3}\\.\\d{1,3}|\\

172\\.1[6-9]{1}\\.\\d{1,3}\\.\\d{1,3}|\\

172\\.2[0-9]{1}\\.\\d{1,3}\\.\\d{1,3}|\\

172\\.3[0-1]{1}\\.\\d{1,3}\\.\\d{1,3} *# regular expression matching trusted IP addresses.*

server.tomcat.max-connections= *# Maximum number of connections that the server will accept and process at any given time.*

server.tomcat.max-http-post-size=0 *# Maximum size in bytes of the HTTP post content.*

server.tomcat.max-threads=0 *# Maximum amount of worker threads.*

server.tomcat.min-spare-threads=0 *# Minimum amount of worker threads.*

server.tomcat.port-header=X-Forwarded-Port *# Name of the HTTP header used to override the original port value.*

server.tomcat.protocol-header= *# Header that holds the incoming protocol, usually named "X-Forwarded-Proto".*

server.tomcat.protocol-header-https-value=https *# Value of the protocol header that indicates that the incoming request uses SSL.*

server.tomcat.redirect-context-root= *# Whether requests to the context root should be redirected by appending a / to the path.*

server.tomcat.remote-ip-header= *# Name of the http header from which the remote ip is extracted. For instance `X-FORWARDED-FOR`*

server.tomcat.uri-encoding=UTF-8 *# Character encoding to use to decode the URI.*

server.undertow.accesslog.dir= *# Undertow access log directory.*

server.undertow.accesslog.enabled=false *# Enable access log.*

server.undertow.accesslog.pattern=common *# Format pattern for access logs.*

server.undertow.accesslog.prefix=access\_log. *# Log file name prefix.*

server.undertow.accesslog.rotate=true *# Enable access log rotation.*

server.undertow.accesslog.suffix=log *# Log file name suffix.*

server.undertow.buffer-size= *# Size of each buffer in bytes.*

server.undertow.buffers-per-region= *# Number of buffer per region.*

server.undertow.direct-buffers= *# Allocate buffers outside the Java heap.*

server.undertow.io-threads= *# Number of I/O threads to create for the worker.*

server.undertow.max-http-post-size=0 *# Maximum size in bytes of the HTTP post content.*

server.undertow.worker-threads= *# Number of worker threads.*

*# FREEMARKER (*[FreeMarkerAutoConfiguration](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/freemarker/FreeMarkerAutoConfiguration.java))

spring.freemarker.allow-request-override=false *# Set whether HttpServletRequest attributes are allowed to override (hide) controller generated model attributes of the same name.*

spring.freemarker.allow-session-override=false *# Set whether HttpSession attributes are allowed to override (hide) controller generated model attributes of the same name.*

spring.freemarker.cache=false *# Enable template caching.*

spring.freemarker.charset=UTF-8 *# Template encoding.*

spring.freemarker.check-template-location=true *# Check that the templates location exists.*

spring.freemarker.content-type=text/html *# Content-Type value.*

spring.freemarker.enabled=true *# Enable MVC view resolution for this technology.*

spring.freemarker.expose-request-attributes=false *# Set whether all request attributes should be added to the model prior to merging with the template.*

spring.freemarker.expose-session-attributes=false *# Set whether all HttpSession attributes should be added to the model prior to merging with the template.*

spring.freemarker.expose-spring-macro-helpers=true *# Set whether to expose a RequestContext for use by Spring's macro library, under the name "springMacroRequestContext".*

spring.freemarker.prefer-file-system-access=true *# Prefer file system access for template loading. File system access enables hot detection of template changes.*

spring.freemarker.prefix= *# Prefix that gets prepended to view names when building a URL.*

spring.freemarker.request-context-attribute= *# Name of the RequestContext attribute for all views.*

spring.freemarker.settings.\*= *# Well-known FreeMarker keys which will be passed to FreeMarker's Configuration.*

spring.freemarker.suffix= *# Suffix that gets appended to view names when building a URL.*

spring.freemarker.template-loader-path=classpath:/templates/ *# Comma-separated list of template paths.*

spring.freemarker.view-names= *# White list of view names that can be resolved.*

*# GROOVY TEMPLATES (*[GroovyTemplateAutoConfiguration](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/groovy/template/GroovyTemplateAutoConfiguration.java))

spring.groovy.template.allow-request-override=false *# Set whether HttpServletRequest attributes are allowed to override (hide) controller generated model attributes of the same name.*

spring.groovy.template.allow-session-override=false *# Set whether HttpSession attributes are allowed to override (hide) controller generated model attributes of the same name.*

spring.groovy.template.cache= *# Enable template caching.*

spring.groovy.template.charset=UTF-8 *# Template encoding.*

spring.groovy.template.check-template-location=true *# Check that the templates location exists.*

spring.groovy.template.configuration.\*= *# See GroovyMarkupConfigurer*

spring.groovy.template.content-type=test/html *# Content-Type value.*

spring.groovy.template.enabled=true *# Enable MVC view resolution for this technology.*

spring.groovy.template.expose-request-attributes=false *# Set whether all request attributes should be added to the model prior to merging with the template.*

spring.groovy.template.expose-session-attributes=false *# Set whether all HttpSession attributes should be added to the model prior to merging with the template.*

spring.groovy.template.expose-spring-macro-helpers=true *# Set whether to expose a RequestContext for use by Spring's macro library, under the name "springMacroRequestContext".*

spring.groovy.template.prefix= *# Prefix that gets prepended to view names when building a URL.*

spring.groovy.template.request-context-attribute= *# Name of the RequestContext attribute for all views.*

spring.groovy.template.resource-loader-path=classpath:/templates/ *# Template path.*

spring.groovy.template.suffix=.tpl *# Suffix that gets appended to view names when building a URL.*

spring.groovy.template.view-names= *# White list of view names that can be resolved.*

*# SPRING HATEOAS (*[HateoasProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/hateoas/HateoasProperties.java))

spring.hateoas.use-hal-as-default-json-media-type=true *# Specify if application/hal+json responses should be sent to requests that accept application/json.*

*# HTTP message conversion*

spring.http.converters.preferred-json-mapper=jackson *# Preferred JSON mapper to use for HTTP message conversion. Set to "gson" to force the use of Gson when both it and Jackson are on the classpath.*

*# HTTP encoding (*[HttpEncodingProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/web/HttpEncodingProperties.java))

spring.http.encoding.charset=UTF-8 *# Charset of HTTP requests and responses. Added to the "Content-Type" header if not set explicitly.*

spring.http.encoding.enabled=true *# Enable http encoding support.*

spring.http.encoding.force= *# Force the encoding to the configured charset on HTTP requests and responses.*

spring.http.encoding.force-request= *# Force the encoding to the configured charset on HTTP requests. Defaults to true when "force" has not been specified.*

spring.http.encoding.force-response= *# Force the encoding to the configured charset on HTTP responses.*

spring.http.encoding.mapping= *# Locale to Encoding mapping.*

*# MULTIPART (*[MultipartProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/web/MultipartProperties.java))

spring.http.multipart.enabled=true *# Enable support of multi-part uploads.*

spring.http.multipart.file-size-threshold=0 *# Threshold after which files will be written to disk. Values can use the suffixed "MB" or "KB" to indicate a Megabyte or Kilobyte size.*

spring.http.multipart.location= *# Intermediate location of uploaded files.*

spring.http.multipart.max-file-size=1MB *# Max file size. Values can use the suffixed "MB" or "KB" to indicate a Megabyte or Kilobyte size.*

spring.http.multipart.max-request-size=10MB *# Max request size. Values can use the suffixed "MB" or "KB" to indicate a Megabyte or Kilobyte size.*

spring.http.multipart.resolve-lazily=false *# Whether to resolve the multipart request lazily at the time of file or parameter access.*

*# JACKSON (*[JacksonProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/jackson/JacksonProperties.java))

spring.jackson.date-format= *# Date format string or a fully-qualified date format class name. For instance `yyyy-MM-dd HH:mm:ss`.*

spring.jackson.default-property-inclusion= *# Controls the inclusion of properties during serialization.*

spring.jackson.deserialization.\*= *# Jackson on/off features that affect the way Java objects are deserialized.*

spring.jackson.generator.\*= *# Jackson on/off features for generators.*

spring.jackson.joda-date-time-format= *# Joda date time format string. If not configured, "date-format" will be used as a fallback if it is configured with a format string.*

spring.jackson.locale= *# Locale used for formatting.*

spring.jackson.mapper.\*= *# Jackson general purpose on/off features.*

spring.jackson.parser.\*= *# Jackson on/off features for parsers.*

spring.jackson.property-naming-strategy= *# One of the constants on Jackson's PropertyNamingStrategy. Can also be a fully-qualified class name of a PropertyNamingStrategy subclass.*

spring.jackson.serialization.\*= *# Jackson on/off features that affect the way Java objects are serialized.*

spring.jackson.time-zone= *# Time zone used when formatting dates. For instance `America/Los\_Angeles`*

*# JERSEY (*[JerseyProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/jersey/JerseyProperties.java))

spring.jersey.application-path= *# Path that serves as the base URI for the application. Overrides the value of "@ApplicationPath" if specified.*

spring.jersey.filter.order=0 *# Jersey filter chain order.*

spring.jersey.init.\*= *# Init parameters to pass to Jersey via the servlet or filter.*

spring.jersey.servlet.load-on-startup=-1 *# Load on startup priority of the Jersey servlet.*

spring.jersey.type=servlet *# Jersey integration type.*

*# SPRING LDAP (*[LdapProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/ldap/LdapProperties.java))

spring.ldap.urls= *# LDAP URLs of the server.*

spring.ldap.base= *# Base suffix from which all operations should originate.*

spring.ldap.username= *# Login user of the server.*

spring.ldap.password= *# Login password of the server.*

spring.ldap.base-environment.\*= *# LDAP specification settings.*

*# EMBEDDED LDAP (*[EmbeddedLdapProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/ldap/embedded/EmbeddedLdapProperties.java))

spring.ldap.embedded.base-dn= *# The base DN*

spring.ldap.embedded.credential.username= *# Embedded LDAP username.*

spring.ldap.embedded.credential.password= *# Embedded LDAP password.*

spring.ldap.embedded.ldif=classpath:schema.ldif *# Schema (LDIF) script resource reference.*

spring.ldap.embedded.port= *# Embedded LDAP port.*

spring.ldap.embedded.validation.enabled=true *# Enable LDAP schema validation.*

spring.ldap.embedded.validation.schema= *# Path to the custom schema.*

*# SPRING MOBILE DEVICE VIEWS (*[DeviceDelegatingViewResolverAutoConfiguration](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/mobile/DeviceDelegatingViewResolverAutoConfiguration.java))

spring.mobile.devicedelegatingviewresolver.enable-fallback=false *# Enable support for fallback resolution.*

spring.mobile.devicedelegatingviewresolver.enabled=false *# Enable device view resolver.*

spring.mobile.devicedelegatingviewresolver.mobile-prefix=mobile/ *# Prefix that gets prepended to view names for mobile devices.*

spring.mobile.devicedelegatingviewresolver.mobile-suffix= *# Suffix that gets appended to view names for mobile devices.*

spring.mobile.devicedelegatingviewresolver.normal-prefix= *# Prefix that gets prepended to view names for normal devices.*

spring.mobile.devicedelegatingviewresolver.normal-suffix= *# Suffix that gets appended to view names for normal devices.*

spring.mobile.devicedelegatingviewresolver.tablet-prefix=tablet/ *# Prefix that gets prepended to view names for tablet devices.*

spring.mobile.devicedelegatingviewresolver.tablet-suffix= *# Suffix that gets appended to view names for tablet devices.*

*# SPRING MOBILE SITE PREFERENCE (*[SitePreferenceAutoConfiguration](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/mobile/SitePreferenceAutoConfiguration.java))

spring.mobile.sitepreference.enabled=true *# Enable SitePreferenceHandler.*

*# MUSTACHE TEMPLATES (*[MustacheAutoConfiguration](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/mustache/MustacheAutoConfiguration.java))

spring.mustache.allow-request-override= *# Set whether HttpServletRequest attributes are allowed to override (hide) controller generated model attributes of the same name.*

spring.mustache.allow-session-override= *# Set whether HttpSession attributes are allowed to override (hide) controller generated model attributes of the same name.*

spring.mustache.cache= *# Enable template caching.*

spring.mustache.charset= *# Template encoding.*

spring.mustache.check-template-location= *# Check that the templates location exists.*

spring.mustache.content-type= *# Content-Type value.*

spring.mustache.enabled= *# Enable MVC view resolution for this technology.*

spring.mustache.expose-request-attributes= *# Set whether all request attributes should be added to the model prior to merging with the template.*

spring.mustache.expose-session-attributes= *# Set whether all HttpSession attributes should be added to the model prior to merging with the template.*

spring.mustache.expose-spring-macro-helpers= *# Set whether to expose a RequestContext for use by Spring's macro library, under the name "springMacroRequestContext".*

spring.mustache.prefix=classpath:/templates/ *# Prefix to apply to template names.*

spring.mustache.request-context-attribute= *# Name of the RequestContext attribute for all views.*

spring.mustache.suffix=.html *# Suffix to apply to template names.*

spring.mustache.view-names= *# White list of view names that can be resolved.*

*# SPRING MVC (*[WebMvcProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/web/WebMvcProperties.java))

spring.mvc.async.request-timeout= *# Amount of time (in milliseconds) before asynchronous request handling times out.*

spring.mvc.date-format= *# Date format to use. For instance `dd/MM/yyyy`.*

spring.mvc.dispatch-trace-request=false *# Dispatch TRACE requests to the FrameworkServlet doService method.*

spring.mvc.dispatch-options-request=true *# Dispatch OPTIONS requests to the FrameworkServlet doService method.*

spring.mvc.favicon.enabled=true *# Enable resolution of favicon.ico.*

spring.mvc.formcontent.putfilter.enabled=true *# Enable Spring's HttpPutFormContentFilter.*

spring.mvc.ignore-default-model-on-redirect=true *# If the content of the "default" model should be ignored during redirect scenarios.*

spring.mvc.locale= *# Locale to use. By default, this locale is overridden by the "Accept-Language" header.*

spring.mvc.locale-resolver=accept-header *# Define how the locale should be resolved.*

spring.mvc.log-resolved-exception=false *# Enable warn logging of exceptions resolved by a "HandlerExceptionResolver".*

spring.mvc.media-types.\*= *# Maps file extensions to media types for content negotiation.*

spring.mvc.message-codes-resolver-format= *# Formatting strategy for message codes. For instance `PREFIX\_ERROR\_CODE`.*

spring.mvc.servlet.load-on-startup=-1 *# Load on startup priority of the Spring Web Services servlet.*

spring.mvc.static-path-pattern=/\*\* *# Path pattern used for static resources.*

spring.mvc.throw-exception-if-no-handler-found=false *# If a "NoHandlerFoundException" should be thrown if no Handler was found to process a request.*

spring.mvc.view.prefix= *# Spring MVC view prefix.*

spring.mvc.view.suffix= *# Spring MVC view suffix.*

*# SPRING RESOURCES HANDLING (*[ResourceProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/web/ResourceProperties.java))

spring.resources.add-mappings=true *# Enable default resource handling.*

spring.resources.cache-period= *# Cache period for the resources served by the resource handler, in seconds.*

spring.resources.chain.cache=true *# Enable caching in the Resource chain.*

spring.resources.chain.enabled= *# Enable the Spring Resource Handling chain. Disabled by default unless at least one strategy has been enabled.*

spring.resources.chain.gzipped=false *# Enable resolution of already gzipped resources.*

spring.resources.chain.html-application-cache=false *# Enable HTML5 application cache manifest rewriting.*

spring.resources.chain.strategy.content.enabled=false *# Enable the content Version Strategy.*

spring.resources.chain.strategy.content.paths=/\*\* *# Comma-separated list of patterns to apply to the Version Strategy.*

spring.resources.chain.strategy.fixed.enabled=false *# Enable the fixed Version Strategy.*

spring.resources.chain.strategy.fixed.paths=/\*\* *# Comma-separated list of patterns to apply to the Version Strategy.*

spring.resources.chain.strategy.fixed.version= *# Version string to use for the Version Strategy.*

spring.resources.static-locations=classpath:/META-INF/resources/,classpath:/resources/,classpath:/static/,classpath:/public/ *# Locations of static resources.*

*# SPRING SESSION (*[SessionProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/session/SessionProperties.java))

spring.session.hazelcast.flush-mode=on-save *# Sessions flush mode.*

spring.session.hazelcast.map-name=spring:session:sessions *# Name of the map used to store sessions.*

spring.session.jdbc.initializer.enabled= *# Create the required session tables on startup if necessary. Enabled automatically if the default table name is set or a custom schema is configured.*

spring.session.jdbc.schema=classpath:org/springframework/session/jdbc/schema-@@platform@@.sql *# Path to the SQL file to use to initialize the database schema.*

spring.session.jdbc.table-name=SPRING\_SESSION *# Name of database table used to store sessions.*

spring.session.mongo.collection-name=sessions *# Collection name used to store sessions.*

spring.session.redis.flush-mode=on-save *# Sessions flush mode.*

spring.session.redis.namespace= *# Namespace for keys used to store sessions.*

spring.session.store-type= *# Session store type.*

*# SPRING SOCIAL (*[SocialWebAutoConfiguration](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/social/SocialWebAutoConfiguration.java))

spring.social.auto-connection-views=false *# Enable the connection status view for supported providers.*

*# SPRING SOCIAL FACEBOOK (*[FacebookAutoConfiguration](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/social/FacebookAutoConfiguration.java))

spring.social.facebook.app-id= *# your application's Facebook App ID*

spring.social.facebook.app-secret= *# your application's Facebook App Secret*

*# SPRING SOCIAL LINKEDIN (*[LinkedInAutoConfiguration](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/social/LinkedInAutoConfiguration.java))

spring.social.linkedin.app-id= *# your application's LinkedIn App ID*

spring.social.linkedin.app-secret= *# your application's LinkedIn App Secret*

*# SPRING SOCIAL TWITTER (*[TwitterAutoConfiguration](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/social/TwitterAutoConfiguration.java))

spring.social.twitter.app-id= *# your application's Twitter App ID*

spring.social.twitter.app-secret= *# your application's Twitter App Secret*

*# THYMELEAF (*[ThymeleafAutoConfiguration](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/thymeleaf/ThymeleafAutoConfiguration.java))

spring.thymeleaf.cache=true *# Enable template caching.*

spring.thymeleaf.check-template=true *# Check that the template exists before rendering it.*

spring.thymeleaf.check-template-location=true *# Check that the templates location exists.*

spring.thymeleaf.content-type=text/html *# Content-Type value.*

spring.thymeleaf.enabled=true *# Enable MVC Thymeleaf view resolution.*

spring.thymeleaf.encoding=UTF-8 *# Template encoding.*

spring.thymeleaf.excluded-view-names= *# Comma-separated list of view names that should be excluded from resolution.*

spring.thymeleaf.mode=HTML5 *# Template mode to be applied to templates. See also StandardTemplateModeHandlers.*

spring.thymeleaf.prefix=classpath:/templates/ *# Prefix that gets prepended to view names when building a URL.*

spring.thymeleaf.suffix=.html *# Suffix that gets appended to view names when building a URL.*

spring.thymeleaf.template-resolver-order= *# Order of the template resolver in the chain.*

spring.thymeleaf.view-names= *# Comma-separated list of view names that can be resolved.*

*# SPRING WEB SERVICES (*[WebServicesProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/webservices/WebServicesProperties.java))

spring.webservices.path=/services *# Path that serves as the base URI for the services.*

spring.webservices.servlet.init= *# Servlet init parameters to pass to Spring Web Services.*

spring.webservices.servlet.load-on-startup=-1 *# Load on startup priority of the Spring Web Services servlet.*

*# ----------------------------------------*

*# SECURITY PROPERTIES*

*# ----------------------------------------*

*# SECURITY (*[SecurityProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/security/SecurityProperties.java))

security.basic.authorize-mode=role *# Security authorize mode to apply.*

security.basic.enabled=true *# Enable basic authentication.*

security.basic.path=/\*\* *# Comma-separated list of paths to secure.*

security.basic.realm=Spring *# HTTP basic realm name.*

security.enable-csrf=false *# Enable Cross Site Request Forgery support.*

security.filter-order=0 *# Security filter chain order.*

security.filter-dispatcher-types=ASYNC, FORWARD, INCLUDE, REQUEST *# Security filter chain dispatcher types.*

security.headers.cache=true *# Enable cache control HTTP headers.*

security.headers.content-security-policy= *# Value for content security policy header.*

security.headers.content-security-policy-mode=default *# Content security policy mode.*

security.headers.content-type=true *# Enable "X-Content-Type-Options" header.*

security.headers.frame=true *# Enable "X-Frame-Options" header.*

security.headers.hsts=all *# HTTP Strict Transport Security (HSTS) mode (none, domain, all).*

security.headers.xss=true *# Enable cross site scripting (XSS) protection.*

security.ignored= *# Comma-separated list of paths to exclude from the default secured paths.*

security.require-ssl=false *# Enable secure channel for all requests.*

security.sessions=stateless *# Session creation policy (always, never, if\_required, stateless).*

security.user.name=user *# Default user name.*

security.user.password= *# Password for the default user name. A random password is logged on startup by default.*

security.user.role=USER *# Granted roles for the default user name.*

*# SECURITY OAUTH2 CLIENT (*[OAuth2ClientProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/security/oauth2/OAuth2ClientProperties.java))

security.oauth2.client.client-id= *# OAuth2 client id.*

security.oauth2.client.client-secret= *# OAuth2 client secret. A random secret is generated by default*

*# SECURITY OAUTH2 RESOURCES (*[ResourceServerProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/security/oauth2/resource/ResourceServerProperties.java))

security.oauth2.resource.filter-order= *# The order of the filter chain used to authenticate tokens.*

security.oauth2.resource.id= *# Identifier of the resource.*

security.oauth2.resource.jwt.key-uri= *# The URI of the JWT token. Can be set if the value is not available and the key is public.*

security.oauth2.resource.jwt.key-value= *# The verification key of the JWT token. Can either be a symmetric secret or PEM-encoded RSA public key.*

security.oauth2.resource.prefer-token-info=true *# Use the token info, can be set to false to use the user info.*

security.oauth2.resource.service-id=resource *#*

security.oauth2.resource.token-info-uri= *# URI of the token decoding endpoint.*

security.oauth2.resource.token-type= *# The token type to send when using the userInfoUri.*

security.oauth2.resource.user-info-uri= *# URI of the user endpoint.*

*# SECURITY OAUTH2 SSO (*[OAuth2SsoProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/security/oauth2/client/OAuth2SsoProperties.java))

security.oauth2.sso.filter-order= *# Filter order to apply if not providing an explicit WebSecurityConfigurerAdapter*

security.oauth2.sso.login-path=/login *# Path to the login page, i.e. the one that triggers the redirect to the OAuth2 Authorization Server*

*# ----------------------------------------*

*# DATA PROPERTIES*

*# ----------------------------------------*

*# FLYWAY (*[FlywayProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/flyway/FlywayProperties.java))

flyway.baseline-description= *#*

flyway.baseline-version=1 *# version to start migration*

flyway.baseline-on-migrate= *#*

flyway.check-location=false *# Check that migration scripts location exists.*

flyway.clean-on-validation-error= *#*

flyway.enabled=true *# Enable flyway.*

flyway.encoding= *#*

flyway.ignore-failed-future-migration= *#*

flyway.init-sqls= *# SQL statements to execute to initialize a connection immediately after obtaining it.*

flyway.locations=classpath:db/migration *# locations of migrations scripts*

flyway.out-of-order= *#*

flyway.password= *# JDBC password if you want Flyway to create its own DataSource*

flyway.placeholder-prefix= *#*

flyway.placeholder-replacement= *#*

flyway.placeholder-suffix= *#*

flyway.placeholders.\*= *#*

flyway.schemas= *# schemas to update*

flyway.sql-migration-prefix=V *#*

flyway.sql-migration-separator= *#*

flyway.sql-migration-suffix=.sql *#*

flyway.table= *#*

flyway.url= *# JDBC url of the database to migrate. If not set, the primary configured data source is used.*

flyway.user= *# Login user of the database to migrate.*

flyway.validate-on-migrate= *#*

*# LIQUIBASE (*[LiquibaseProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/liquibase/LiquibaseProperties.java))

liquibase.change-log=classpath:/db/changelog/db.changelog-master.yaml *# Change log configuration path.*

liquibase.check-change-log-location=true *# Check the change log location exists.*

liquibase.contexts= *# Comma-separated list of runtime contexts to use.*

liquibase.default-schema= *# Default database schema.*

liquibase.drop-first=false *# Drop the database schema first.*

liquibase.enabled=true *# Enable liquibase support.*

liquibase.labels= *# Comma-separated list of runtime labels to use.*

liquibase.parameters.\*= *# Change log parameters.*

liquibase.password= *# Login password of the database to migrate.*

liquibase.rollback-file= *# File to which rollback SQL will be written when an update is performed.*

liquibase.url= *# JDBC url of the database to migrate. If not set, the primary configured data source is used.*

liquibase.user= *# Login user of the database to migrate.*

*# COUCHBASE (*[CouchbaseProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/couchbase/CouchbaseProperties.java))

spring.couchbase.bootstrap-hosts= *# Couchbase nodes (host or IP address) to bootstrap from.*

spring.couchbase.bucket.name=default *# Name of the bucket to connect to.*

spring.couchbase.bucket.password= *# Password of the bucket.*

spring.couchbase.env.endpoints.key-value=1 *# Number of sockets per node against the Key/value service.*

spring.couchbase.env.endpoints.query=1 *# Number of sockets per node against the Query (N1QL) service.*

spring.couchbase.env.endpoints.view=1 *# Number of sockets per node against the view service.*

spring.couchbase.env.ssl.enabled= *# Enable SSL support. Enabled automatically if a "keyStore" is provided unless specified otherwise.*

spring.couchbase.env.ssl.key-store= *# Path to the JVM key store that holds the certificates.*

spring.couchbase.env.ssl.key-store-password= *# Password used to access the key store.*

spring.couchbase.env.timeouts.connect=5000 *# Bucket connections timeout in milliseconds.*

spring.couchbase.env.timeouts.key-value=2500 *# Blocking operations performed on a specific key timeout in milliseconds.*

spring.couchbase.env.timeouts.query=7500 *# N1QL query operations timeout in milliseconds.*

spring.couchbase.env.timeouts.socket-connect=1000 *# Socket connect connections timeout in milliseconds.*

spring.couchbase.env.timeouts.view=7500 *# Regular and geospatial view operations timeout in milliseconds.*

*# DAO (*[PersistenceExceptionTranslationAutoConfiguration](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/dao/PersistenceExceptionTranslationAutoConfiguration.java))

spring.dao.exceptiontranslation.enabled=true *# Enable the PersistenceExceptionTranslationPostProcessor.*

*# CASSANDRA (*[CassandraProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/cassandra/CassandraProperties.java))

spring.data.cassandra.cluster-name= *# Name of the Cassandra cluster.*

spring.data.cassandra.compression=none *# Compression supported by the Cassandra binary protocol.*

spring.data.cassandra.connect-timeout-millis= *# Socket option: connection time out.*

spring.data.cassandra.consistency-level= *# Queries consistency level.*

spring.data.cassandra.contact-points=localhost *# Comma-separated list of cluster node addresses.*

spring.data.cassandra.fetch-size= *# Queries default fetch size.*

spring.data.cassandra.keyspace-name= *# Keyspace name to use.*

spring.data.cassandra.load-balancing-policy= *# Class name of the load balancing policy.*

spring.data.cassandra.port= *# Port of the Cassandra server.*

spring.data.cassandra.password= *# Login password of the server.*

spring.data.cassandra.read-timeout-millis= *# Socket option: read time out.*

spring.data.cassandra.reconnection-policy= *# Reconnection policy class.*

spring.data.cassandra.retry-policy= *# Class name of the retry policy.*

spring.data.cassandra.serial-consistency-level= *# Queries serial consistency level.*

spring.data.cassandra.schema-action=none *# Schema action to take at startup.*

spring.data.cassandra.ssl=false *# Enable SSL support.*

spring.data.cassandra.username= *# Login user of the server.*

*# DATA COUCHBASE (*[CouchbaseDataProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/data/couchbase/CouchbaseDataProperties.java))

spring.data.couchbase.auto-index=false *# Automatically create views and indexes.*

spring.data.couchbase.consistency=read-your-own-writes *# Consistency to apply by default on generated queries.*

spring.data.couchbase.repositories.enabled=true *# Enable Couchbase repositories.*

*# ELASTICSEARCH (*[ElasticsearchProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/data/elasticsearch/ElasticsearchProperties.java))

spring.data.elasticsearch.cluster-name=elasticsearch *# Elasticsearch cluster name.*

spring.data.elasticsearch.cluster-nodes= *# Comma-separated list of cluster node addresses. If not specified, starts a client node.*

spring.data.elasticsearch.properties.\*= *# Additional properties used to configure the client.*

spring.data.elasticsearch.repositories.enabled=true *# Enable Elasticsearch repositories.*

*# DATA LDAP*

spring.data.ldap.repositories.enabled=true *# Enable LDAP repositories.*

*# MONGODB (*[MongoProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/mongo/MongoProperties.java))

spring.data.mongodb.authentication-database= *# Authentication database name.*

spring.data.mongodb.database=test *# Database name.*

spring.data.mongodb.field-naming-strategy= *# Fully qualified name of the FieldNamingStrategy to use.*

spring.data.mongodb.grid-fs-database= *# GridFS database name.*

spring.data.mongodb.host=localhost *# Mongo server host. Cannot be set with uri.*

spring.data.mongodb.password= *# Login password of the mongo server. Cannot be set with uri.*

spring.data.mongodb.port=27017 *# Mongo server port. Cannot be set with uri.*

spring.data.mongodb.repositories.enabled=true *# Enable Mongo repositories.*

spring.data.mongodb.uri=mongodb://localhost/test *# Mongo database URI. Cannot be set with host, port and credentials.*

spring.data.mongodb.username= *# Login user of the mongo server. Cannot be set with uri.*

*# DATA REDIS*

spring.data.redis.repositories.enabled=true *# Enable Redis repositories.*

*# NEO4J (*[Neo4jProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/neo4j/Neo4jProperties.java))

spring.data.neo4j.compiler= *# Compiler to use.*

spring.data.neo4j.embedded.enabled=true *# Enable embedded mode if the embedded driver is available.*

spring.data.neo4j.open-in-view=false *# Register OpenSessionInViewInterceptor. Binds a Neo4j Session to the thread for the entire processing of the request.*

spring.data.neo4j.password= *# Login password of the server.*

spring.data.neo4j.repositories.enabled=true *# Enable Neo4j repositories.*

spring.data.neo4j.uri= *# URI used by the driver. Auto-detected by default.*

spring.data.neo4j.username= *# Login user of the server.*

*# DATA REST (*[RepositoryRestProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/data/rest/RepositoryRestProperties.java))

spring.data.rest.base-path= *# Base path to be used by Spring Data REST to expose repository resources.*

spring.data.rest.default-page-size= *# Default size of pages.*

spring.data.rest.detection-strategy=default *# Strategy to use to determine which repositories get exposed.*

spring.data.rest.enable-enum-translation= *# Enable enum value translation via the Spring Data REST default resource bundle.*

spring.data.rest.limit-param-name= *# Name of the URL query string parameter that indicates how many results to return at once.*

spring.data.rest.max-page-size= *# Maximum size of pages.*

spring.data.rest.page-param-name= *# Name of the URL query string parameter that indicates what page to return.*

spring.data.rest.return-body-on-create= *# Return a response body after creating an entity.*

spring.data.rest.return-body-on-update= *# Return a response body after updating an entity.*

spring.data.rest.sort-param-name= *# Name of the URL query string parameter that indicates what direction to sort results.*

*# SOLR (*[SolrProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/solr/SolrProperties.java))

spring.data.solr.host=http://127.0.0.1:8983/solr *# Solr host. Ignored if "zk-host" is set.*

spring.data.solr.repositories.enabled=true *# Enable Solr repositories.*

spring.data.solr.zk-host= *# ZooKeeper host address in the form HOST:PORT.*

*# DATASOURCE (*[DataSourceAutoConfiguration](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/jdbc/DataSourceAutoConfiguration.java) & [DataSourceProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/jdbc/DataSourceProperties.java))

spring.datasource.continue-on-error=false *# Do not stop if an error occurs while initializing the database.*

spring.datasource.data= *# Data (DML) script resource references.*

spring.datasource.data-username= *# User of the database to execute DML scripts (if different).*

spring.datasource.data-password= *# Password of the database to execute DML scripts (if different).*

spring.datasource.dbcp2.\*= *# Commons DBCP2 specific settings*

spring.datasource.driver-class-name= *# Fully qualified name of the JDBC driver. Auto-detected based on the URL by default.*

spring.datasource.generate-unique-name=false *# Generate a random datasource name.*

spring.datasource.hikari.\*= *# Hikari specific settings*

spring.datasource.initialize=true *# Populate the database using 'data.sql'.*

spring.datasource.jmx-enabled=false *# Enable JMX support (if provided by the underlying pool).*

spring.datasource.jndi-name= *# JNDI location of the datasource. Class, url, username & password are ignored when set.*

spring.datasource.name=testdb *# Name of the datasource.*

spring.datasource.password= *# Login password of the database.*

spring.datasource.platform=all *# Platform to use in the schema resource (schema-${platform}.sql).*

spring.datasource.schema= *# Schema (DDL) script resource references.*

spring.datasource.schema-username= *# User of the database to execute DDL scripts (if different).*

spring.datasource.schema-password= *# Password of the database to execute DDL scripts (if different).*

spring.datasource.separator=; *# Statement separator in SQL initialization scripts.*

spring.datasource.sql-script-encoding= *# SQL scripts encoding.*

spring.datasource.tomcat.\*= *# Tomcat datasource specific settings*

spring.datasource.type= *# Fully qualified name of the connection pool implementation to use. By default, it is auto-detected from the classpath.*

spring.datasource.url= *# JDBC url of the database.*

spring.datasource.username=

*# JEST (Elasticsearch HTTP client) (*[JestProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/jest/JestProperties.java))

spring.elasticsearch.jest.connection-timeout=3000 *# Connection timeout in milliseconds.*

spring.elasticsearch.jest.multi-threaded=true *# Enable connection requests from multiple execution threads.*

spring.elasticsearch.jest.password= *# Login password.*

spring.elasticsearch.jest.proxy.host= *# Proxy host the HTTP client should use.*

spring.elasticsearch.jest.proxy.port= *# Proxy port the HTTP client should use.*

spring.elasticsearch.jest.read-timeout=3000 *# Read timeout in milliseconds.*

spring.elasticsearch.jest.uris=http://localhost:9200 *# Comma-separated list of the Elasticsearch instances to use.*

spring.elasticsearch.jest.username= *# Login user.*

*# H2 Web Console (*[H2ConsoleProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/h2/H2ConsoleProperties.java))

spring.h2.console.enabled=false *# Enable the console.*

spring.h2.console.path=/h2-console *# Path at which the console will be available.*

spring.h2.console.settings.trace=false *# Enable trace output.*

spring.h2.console.settings.web-allow-others=false *# Enable remote access.*

*# JOOQ (*[JooqAutoConfiguration](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/jooq/JooqAutoConfiguration.java))

spring.jooq.sql-dialect= *# SQLDialect JOOQ used when communicating with the configured datasource. For instance `POSTGRES`*

*# JPA (*[JpaBaseConfiguration](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/orm/jpa/JpaBaseConfiguration.java), [HibernateJpaAutoConfiguration](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/orm/jpa/HibernateJpaAutoConfiguration.java))

spring.data.jpa.repositories.enabled=true *# Enable JPA repositories.*

spring.jpa.database= *# Target database to operate on, auto-detected by default. Can be alternatively set using the "databasePlatform" property.*

spring.jpa.database-platform= *# Name of the target database to operate on, auto-detected by default. Can be alternatively set using the "Database" enum.*

spring.jpa.generate-ddl=false *# Initialize the schema on startup.*

spring.jpa.hibernate.ddl-auto= *# DDL mode. This is actually a shortcut for the "hibernate.hbm2ddl.auto" property. Default to "create-drop" when using an embedded database, "none" otherwise.*

spring.jpa.hibernate.naming.implicit-strategy= *# Hibernate 5 implicit naming strategy fully qualified name.*

spring.jpa.hibernate.naming.physical-strategy= *# Hibernate 5 physical naming strategy fully qualified name.*

spring.jpa.hibernate.naming.strategy= *# Hibernate 4 naming strategy fully qualified name. Not supported with Hibernate 5.*

spring.jpa.hibernate.use-new-id-generator-mappings= *# Use Hibernate's newer IdentifierGenerator for AUTO, TABLE and SEQUENCE.*

spring.jpa.open-in-view=true *# Register OpenEntityManagerInViewInterceptor. Binds a JPA EntityManager to the thread for the entire processing of the request.*

spring.jpa.properties.\*= *# Additional native properties to set on the JPA provider.*

spring.jpa.show-sql=false *# Enable logging of SQL statements.*

*# JTA (*[JtaAutoConfiguration](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/transaction/jta/JtaAutoConfiguration.java))

spring.jta.enabled=true *# Enable JTA support.*

spring.jta.log-dir= *# Transaction logs directory.*

spring.jta.transaction-manager-id= *# Transaction manager unique identifier.*

*# ATOMIKOS (*[AtomikosProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot/src/main/java/org/springframework/boot/jta/atomikos/AtomikosProperties.java))

spring.jta.atomikos.connectionfactory.borrow-connection-timeout=30 *# Timeout, in seconds, for borrowing connections from the pool.*

spring.jta.atomikos.connectionfactory.ignore-session-transacted-flag=true *# Whether or not to ignore the transacted flag when creating session.*

spring.jta.atomikos.connectionfactory.local-transaction-mode=false *# Whether or not local transactions are desired.*

spring.jta.atomikos.connectionfactory.maintenance-interval=60 *# The time, in seconds, between runs of the pool's maintenance thread.*

spring.jta.atomikos.connectionfactory.max-idle-time=60 *# The time, in seconds, after which connections are cleaned up from the pool.*

spring.jta.atomikos.connectionfactory.max-lifetime=0 *# The time, in seconds, that a connection can be pooled for before being destroyed. 0 denotes no limit.*

spring.jta.atomikos.connectionfactory.max-pool-size=1 *# The maximum size of the pool.*

spring.jta.atomikos.connectionfactory.min-pool-size=1 *# The minimum size of the pool.*

spring.jta.atomikos.connectionfactory.reap-timeout=0 *# The reap timeout, in seconds, for borrowed connections. 0 denotes no limit.*

spring.jta.atomikos.connectionfactory.unique-resource-name=jmsConnectionFactory *# The unique name used to identify the resource during recovery.*

spring.jta.atomikos.datasource.borrow-connection-timeout=30 *# Timeout, in seconds, for borrowing connections from the pool.*

spring.jta.atomikos.datasource.default-isolation-level= *# Default isolation level of connections provided by the pool.*

spring.jta.atomikos.datasource.login-timeout= *# Timeout, in seconds, for establishing a database connection.*

spring.jta.atomikos.datasource.maintenance-interval=60 *# The time, in seconds, between runs of the pool's maintenance thread.*

spring.jta.atomikos.datasource.max-idle-time=60 *# The time, in seconds, after which connections are cleaned up from the pool.*

spring.jta.atomikos.datasource.max-lifetime=0 *# The time, in seconds, that a connection can be pooled for before being destroyed. 0 denotes no limit.*

spring.jta.atomikos.datasource.max-pool-size=1 *# The maximum size of the pool.*

spring.jta.atomikos.datasource.min-pool-size=1 *# The minimum size of the pool.*

spring.jta.atomikos.datasource.reap-timeout=0 *# The reap timeout, in seconds, for borrowed connections. 0 denotes no limit.*

spring.jta.atomikos.datasource.test-query= *# SQL query or statement used to validate a connection before returning it.*

spring.jta.atomikos.datasource.unique-resource-name=dataSource *# The unique name used to identify the resource during recovery.*

spring.jta.atomikos.properties.checkpoint-interval=500 *# Interval between checkpoints.*

spring.jta.atomikos.properties.console-file-count=1 *# Number of debug logs files that can be created.*

spring.jta.atomikos.properties.console-file-limit=-1 *# How many bytes can be stored at most in debug logs files.*

spring.jta.atomikos.properties.console-file-name=tm.out *# Debug logs file name.*

spring.jta.atomikos.properties.console-log-level=warn *# Console log level.*

spring.jta.atomikos.properties.default-jta-timeout=10000 *# Default timeout for JTA transactions.*

spring.jta.atomikos.properties.enable-logging=true *# Enable disk logging.*

spring.jta.atomikos.properties.force-shutdown-on-vm-exit=false *# Specify if a VM shutdown should trigger forced shutdown of the transaction core.*

spring.jta.atomikos.properties.log-base-dir= *# Directory in which the log files should be stored.*

spring.jta.atomikos.properties.log-base-name=tmlog *# Transactions log file base name.*

spring.jta.atomikos.properties.max-actives=50 *# Maximum number of active transactions.*

spring.jta.atomikos.properties.max-timeout=300000 *# Maximum timeout (in milliseconds) that can be allowed for transactions.*

spring.jta.atomikos.properties.output-dir= *# Directory in which to store the debug log files.*

spring.jta.atomikos.properties.serial-jta-transactions=true *# Specify if sub-transactions should be joined when possible.*

spring.jta.atomikos.properties.service= *# Transaction manager implementation that should be started.*

spring.jta.atomikos.properties.threaded-two-phase-commit=true *# Use different (and concurrent) threads for two-phase commit on the participating resources.*

spring.jta.atomikos.properties.transaction-manager-unique-name= *# Transaction manager's unique name.*

*# BITRONIX*

spring.jta.bitronix.connectionfactory.acquire-increment=1 *# Number of connections to create when growing the pool.*

spring.jta.bitronix.connectionfactory.acquisition-interval=1 *# Time, in seconds, to wait before trying to acquire a connection again after an invalid connection was acquired.*

spring.jta.bitronix.connectionfactory.acquisition-timeout=30 *# Timeout, in seconds, for acquiring connections from the pool.*

spring.jta.bitronix.connectionfactory.allow-local-transactions=true *# Whether or not the transaction manager should allow mixing XA and non-XA transactions.*

spring.jta.bitronix.connectionfactory.apply-transaction-timeout=false *# Whether or not the transaction timeout should be set on the XAResource when it is enlisted.*

spring.jta.bitronix.connectionfactory.automatic-enlisting-enabled=true *# Whether or not resources should be enlisted and delisted automatically.*

spring.jta.bitronix.connectionfactory.cache-producers-consumers=true *# Whether or not produces and consumers should be cached.*

spring.jta.bitronix.connectionfactory.defer-connection-release=true *# Whether or not the provider can run many transactions on the same connection and supports transaction interleaving.*

spring.jta.bitronix.connectionfactory.ignore-recovery-failures=false *# Whether or not recovery failures should be ignored.*

spring.jta.bitronix.connectionfactory.max-idle-time=60 *# The time, in seconds, after which connections are cleaned up from the pool.*

spring.jta.bitronix.connectionfactory.max-pool-size=10 *# The maximum size of the pool. 0 denotes no limit.*

spring.jta.bitronix.connectionfactory.min-pool-size=0 *# The minimum size of the pool.*

spring.jta.bitronix.connectionfactory.password= *# The password to use to connect to the JMS provider.*

spring.jta.bitronix.connectionfactory.share-transaction-connections=false *# Whether or not connections in the ACCESSIBLE state can be shared within the context of a transaction.*

spring.jta.bitronix.connectionfactory.test-connections=true *# Whether or not connections should be tested when acquired from the pool.*

spring.jta.bitronix.connectionfactory.two-pc-ordering-position=1 *# The position that this resource should take during two-phase commit (always first is Integer.MIN\_VALUE, always last is Integer.MAX\_VALUE).*

spring.jta.bitronix.connectionfactory.unique-name=jmsConnectionFactory *# The unique name used to identify the resource during recovery.*

spring.jta.bitronix.connectionfactory.use-tm-join=true Whether or not TMJOIN should be used when starting XAResources.

spring.jta.bitronix.connectionfactory.user= *# The user to use to connect to the JMS provider.*

spring.jta.bitronix.datasource.acquire-increment=1 *# Number of connections to create when growing the pool.*

spring.jta.bitronix.datasource.acquisition-interval=1 *# Time, in seconds, to wait before trying to acquire a connection again after an invalid connection was acquired.*

spring.jta.bitronix.datasource.acquisition-timeout=30 *# Timeout, in seconds, for acquiring connections from the pool.*

spring.jta.bitronix.datasource.allow-local-transactions=true *# Whether or not the transaction manager should allow mixing XA and non-XA transactions.*

spring.jta.bitronix.datasource.apply-transaction-timeout=false *# Whether or not the transaction timeout should be set on the XAResource when it is enlisted.*

spring.jta.bitronix.datasource.automatic-enlisting-enabled=true *# Whether or not resources should be enlisted and delisted automatically.*

spring.jta.bitronix.datasource.cursor-holdability= *# The default cursor holdability for connections.*

spring.jta.bitronix.datasource.defer-connection-release=true *# Whether or not the database can run many transactions on the same connection and supports transaction interleaving.*

spring.jta.bitronix.datasource.enable-jdbc4-connection-test= *# Whether or not Connection.isValid() is called when acquiring a connection from the pool.*

spring.jta.bitronix.datasource.ignore-recovery-failures=false *# Whether or not recovery failures should be ignored.*

spring.jta.bitronix.datasource.isolation-level= *# The default isolation level for connections.*

spring.jta.bitronix.datasource.local-auto-commit= *# The default auto-commit mode for local transactions.*

spring.jta.bitronix.datasource.login-timeout= *# Timeout, in seconds, for establishing a database connection.*

spring.jta.bitronix.datasource.max-idle-time=60 *# The time, in seconds, after which connections are cleaned up from the pool.*

spring.jta.bitronix.datasource.max-pool-size=10 *# The maximum size of the pool. 0 denotes no limit.*

spring.jta.bitronix.datasource.min-pool-size=0 *# The minimum size of the pool.*

spring.jta.bitronix.datasource.prepared-statement-cache-size=0 *# The target size of the prepared statement cache. 0 disables the cache.*

spring.jta.bitronix.datasource.share-transaction-connections=false *# Whether or not connections in the ACCESSIBLE state can be shared within the context of a transaction.*

spring.jta.bitronix.datasource.test-query= *# SQL query or statement used to validate a connection before returning it.*

spring.jta.bitronix.datasource.two-pc-ordering-position=1 *# The position that this resource should take during two-phase commit (always first is Integer.MIN\_VALUE, always last is Integer.MAX\_VALUE).*

spring.jta.bitronix.datasource.unique-name=dataSource *# The unique name used to identify the resource during recovery.*

spring.jta.bitronix.datasource.use-tm-join=true Whether or not TMJOIN should be used when starting XAResources.

spring.jta.bitronix.properties.allow-multiple-lrc=false *# Allow multiple LRC resources to be enlisted into the same transaction.*

spring.jta.bitronix.properties.asynchronous2-pc=false *# Enable asynchronously execution of two phase commit.*

spring.jta.bitronix.properties.background-recovery-interval-seconds=60 *# Interval in seconds at which to run the recovery process in the background.*

spring.jta.bitronix.properties.current-node-only-recovery=true *# Recover only the current node.*

spring.jta.bitronix.properties.debug-zero-resource-transaction=false *# Log the creation and commit call stacks of transactions executed without a single enlisted resource.*

spring.jta.bitronix.properties.default-transaction-timeout=60 *# Default transaction timeout in seconds.*

spring.jta.bitronix.properties.disable-jmx=false *# Enable JMX support.*

spring.jta.bitronix.properties.exception-analyzer= *# Set the fully qualified name of the exception analyzer implementation to use.*

spring.jta.bitronix.properties.filter-log-status=false *# Enable filtering of logs so that only mandatory logs are written.*

spring.jta.bitronix.properties.force-batching-enabled=true *# Set if disk forces are batched.*

spring.jta.bitronix.properties.forced-write-enabled=true *# Set if logs are forced to disk.*

spring.jta.bitronix.properties.graceful-shutdown-interval=60 *# Maximum amount of seconds the TM will wait for transactions to get done before aborting them at shutdown time.*

spring.jta.bitronix.properties.jndi-transaction-synchronization-registry-name= *# JNDI name of the TransactionSynchronizationRegistry.*

spring.jta.bitronix.properties.jndi-user-transaction-name= *# JNDI name of the UserTransaction.*

spring.jta.bitronix.properties.journal=disk *# Name of the journal. Can be 'disk', 'null' or a class name.*

spring.jta.bitronix.properties.log-part1-filename=btm1.tlog *# Name of the first fragment of the journal.*

spring.jta.bitronix.properties.log-part2-filename=btm2.tlog *# Name of the second fragment of the journal.*

spring.jta.bitronix.properties.max-log-size-in-mb=2 *# Maximum size in megabytes of the journal fragments.*

spring.jta.bitronix.properties.resource-configuration-filename= *# ResourceLoader configuration file name.*

spring.jta.bitronix.properties.server-id= *# ASCII ID that must uniquely identify this TM instance. Default to the machine's IP address.*

spring.jta.bitronix.properties.skip-corrupted-logs=false *# Skip corrupted transactions log entries.*

spring.jta.bitronix.properties.warn-about-zero-resource-transaction=true *# Log a warning for transactions executed without a single enlisted resource.*

*# NARAYANA (*[NarayanaProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot/src/main/java/org/springframework/boot/jta/narayana/NarayanaProperties.java))

spring.jta.narayana.default-timeout=60 *# Transaction timeout in seconds.*

spring.jta.narayana.expiry-scanners=com.arjuna.ats.internal.arjuna.recovery.ExpiredTransactionStatusManagerScanner *# Comma-separated list of expiry scanners.*

spring.jta.narayana.log-dir= *# Transaction object store directory.*

spring.jta.narayana.one-phase-commit=true *# Enable one phase commit optimisation.*

spring.jta.narayana.periodic-recovery-period=120 *# Interval in which periodic recovery scans are performed in seconds.*

spring.jta.narayana.recovery-backoff-period=10 *# Back off period between first and second phases of the recovery scan in seconds.*

spring.jta.narayana.recovery-db-pass= *# Database password to be used by recovery manager.*

spring.jta.narayana.recovery-db-user= *# Database username to be used by recovery manager.*

spring.jta.narayana.recovery-jms-pass= *# JMS password to be used by recovery manager.*

spring.jta.narayana.recovery-jms-user= *# JMS username to be used by recovery manager.*

spring.jta.narayana.recovery-modules= *# Comma-separated list of recovery modules.*

spring.jta.narayana.transaction-manager-id=1 *# Unique transaction manager id.*

spring.jta.narayana.xa-resource-orphan-filters= *# Comma-separated list of orphan filters.*

*# EMBEDDED MONGODB (*[EmbeddedMongoProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/mongo/embedded/EmbeddedMongoProperties.java))

spring.mongodb.embedded.features=SYNC\_DELAY *# Comma-separated list of features to enable.*

spring.mongodb.embedded.storage.database-dir= *# Directory used for data storage.*

spring.mongodb.embedded.storage.oplog-size= *# Maximum size of the oplog in megabytes.*

spring.mongodb.embedded.storage.repl-set-name= *# Name of the replica set.*

spring.mongodb.embedded.version=2.6.10 *# Version of Mongo to use.*

*# REDIS (*[RedisProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/data/redis/RedisProperties.java))

spring.redis.cluster.max-redirects= *# Maximum number of redirects to follow when executing commands across the cluster.*

spring.redis.cluster.nodes= *# Comma-separated list of "host:port" pairs to bootstrap from.*

spring.redis.database=0 *# Database index used by the connection factory.*

spring.redis.url= *# Connection URL, will override host, port and password (user will be ignored), e.g. redis://user:password@example.com:6379*

spring.redis.host=localhost *# Redis server host.*

spring.redis.password= *# Login password of the redis server.*

spring.redis.ssl=false *# Enable SSL support.*

spring.redis.pool.max-active=8 *# Max number of connections that can be allocated by the pool at a given time. Use a negative value for no limit.*

spring.redis.pool.max-idle=8 *# Max number of "idle" connections in the pool. Use a negative value to indicate an unlimited number of idle connections.*

spring.redis.pool.max-wait=-1 *# Maximum amount of time (in milliseconds) a connection allocation should block before throwing an exception when the pool is exhausted. Use a negative value to block indefinitely.*

spring.redis.pool.min-idle=0 *# Target for the minimum number of idle connections to maintain in the pool. This setting only has an effect if it is positive.*

spring.redis.port=6379 *# Redis server port.*

spring.redis.sentinel.master= *# Name of Redis server.*

spring.redis.sentinel.nodes= *# Comma-separated list of host:port pairs.*

spring.redis.timeout=0 *# Connection timeout in milliseconds.*

*# TRANSACTION (*[TransactionProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/transaction/TransactionProperties.java))

spring.transaction.default-timeout= *# Default transaction timeout in seconds.*

spring.transaction.rollback-on-commit-failure= *# Perform the rollback on commit failures.*

*# ----------------------------------------*

*# INTEGRATION PROPERTIES*

*# ----------------------------------------*

*# ACTIVEMQ (*[ActiveMQProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/jms/activemq/ActiveMQProperties.java))

spring.activemq.broker-url= *# URL of the ActiveMQ broker. Auto-generated by default. For instance `tcp://localhost:61616`*

spring.activemq.in-memory=true *# Specify if the default broker URL should be in memory. Ignored if an explicit broker has been specified.*

spring.activemq.password= *# Login password of the broker.*

spring.activemq.user= *# Login user of the broker.*

spring.activemq.packages.trust-all=false *# Trust all packages.*

spring.activemq.packages.trusted= *# Comma-separated list of specific packages to trust (when not trusting all packages).*

spring.activemq.pool.configuration.\*= *# See PooledConnectionFactory.*

spring.activemq.pool.enabled=false *# Whether a PooledConnectionFactory should be created instead of a regular ConnectionFactory.*

spring.activemq.pool.expiry-timeout=0 *# Connection expiration timeout in milliseconds.*

spring.activemq.pool.idle-timeout=30000 *# Connection idle timeout in milliseconds.*

spring.activemq.pool.max-connections=1 *# Maximum number of pooled connections.*

*# ARTEMIS (*[ArtemisProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/jms/artemis/ArtemisProperties.java))

spring.artemis.embedded.cluster-password= *# Cluster password. Randomly generated on startup by default.*

spring.artemis.embedded.data-directory= *# Journal file directory. Not necessary if persistence is turned off.*

spring.artemis.embedded.enabled=true *# Enable embedded mode if the Artemis server APIs are available.*

spring.artemis.embedded.persistent=false *# Enable persistent store.*

spring.artemis.embedded.queues= *# Comma-separated list of queues to create on startup.*

spring.artemis.embedded.server-id= *# Server id. By default, an auto-incremented counter is used.*

spring.artemis.embedded.topics= *# Comma-separated list of topics to create on startup.*

spring.artemis.host=localhost *# Artemis broker host.*

spring.artemis.mode= *# Artemis deployment mode, auto-detected by default.*

spring.artemis.password= *# Login password of the broker.*

spring.artemis.port=61616 *# Artemis broker port.*

spring.artemis.user= *# Login user of the broker.*

*# SPRING BATCH (*[BatchProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/batch/BatchProperties.java))

spring.batch.initializer.enabled= *# Create the required batch tables on startup if necessary. Enabled automatically if no custom table prefix is set or if a custom schema is configured.*

spring.batch.job.enabled=true *# Execute all Spring Batch jobs in the context on startup.*

spring.batch.job.names= *# Comma-separated list of job names to execute on startup (For instance `job1,job2`). By default, all Jobs found in the context are executed.*

spring.batch.schema=classpath:org/springframework/batch/core/schema-@@platform@@.sql *# Path to the SQL file to use to initialize the database schema.*

spring.batch.table-prefix= *# Table prefix for all the batch meta-data tables.*

*# JMS (*[JmsProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/jms/JmsProperties.java))

spring.jms.jndi-name= *# Connection factory JNDI name. When set, takes precedence to others connection factory auto-configurations.*

spring.jms.listener.acknowledge-mode= *# Acknowledge mode of the container. By default, the listener is transacted with automatic acknowledgment.*

spring.jms.listener.auto-startup=true *# Start the container automatically on startup.*

spring.jms.listener.concurrency= *# Minimum number of concurrent consumers.*

spring.jms.listener.max-concurrency= *# Maximum number of concurrent consumers.*

spring.jms.pub-sub-domain=false *# Specify if the default destination type is topic.*

spring.jms.template.default-destination= *# Default destination to use on send/receive operations that do not have a destination parameter.*

spring.jms.template.delivery-delay= *# Delivery delay to use for send calls in milliseconds.*

spring.jms.template.delivery-mode= *# Delivery mode. Enable QoS when set.*

spring.jms.template.priority= *# Priority of a message when sending. Enable QoS when set.*

spring.jms.template.qos-enabled= *# Enable explicit QoS when sending a message.*

spring.jms.template.receive-timeout= *# Timeout to use for receive calls in milliseconds.*

spring.jms.template.time-to-live= *# Time-to-live of a message when sending in milliseconds. Enable QoS when set.*

*# APACHE KAFKA (*[KafkaProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/kafka/KafkaProperties.java))

spring.kafka.bootstrap-servers= *# Comma-delimited list of host:port pairs to use for establishing the initial connection to the Kafka cluster.*

spring.kafka.client-id= *# Id to pass to the server when making requests; used for server-side logging.*

spring.kafka.consumer.auto-commit-interval= *# Frequency in milliseconds that the consumer offsets are auto-committed to Kafka if 'enable.auto.commit' true.*

spring.kafka.consumer.auto-offset-reset= *# What to do when there is no initial offset in Kafka or if the current offset does not exist any more on the server.*

spring.kafka.consumer.bootstrap-servers= *# Comma-delimited list of host:port pairs to use for establishing the initial connection to the Kafka cluster.*

spring.kafka.consumer.client-id= *# Id to pass to the server when making requests; used for server-side logging.*

spring.kafka.consumer.enable-auto-commit= *# If true the consumer's offset will be periodically committed in the background.*

spring.kafka.consumer.fetch-max-wait= *# Maximum amount of time in milliseconds the server will block before answering the fetch request if there isn't sufficient data to immediately satisfy the requirement given by "fetch.min.bytes".*

spring.kafka.consumer.fetch-min-size= *# Minimum amount of data the server should return for a fetch request in bytes.*

spring.kafka.consumer.group-id= *# Unique string that identifies the consumer group this consumer belongs to.*

spring.kafka.consumer.heartbeat-interval= *# Expected time in milliseconds between heartbeats to the consumer coordinator.*

spring.kafka.consumer.key-deserializer= *# Deserializer class for keys.*

spring.kafka.consumer.max-poll-records= *# Maximum number of records returned in a single call to poll().*

spring.kafka.consumer.value-deserializer= *# Deserializer class for values.*

spring.kafka.listener.ack-count= *# Number of records between offset commits when ackMode is "COUNT" or "COUNT\_TIME".*

spring.kafka.listener.ack-mode= *# Listener AckMode; see the spring-kafka documentation.*

spring.kafka.listener.ack-time= *# Time in milliseconds between offset commits when ackMode is "TIME" or "COUNT\_TIME".*

spring.kafka.listener.concurrency= *# Number of threads to run in the listener containers.*

spring.kafka.listener.poll-timeout= *# Timeout in milliseconds to use when polling the consumer.*

spring.kafka.producer.acks= *# Number of acknowledgments the producer requires the leader to have received before considering a request complete.*

spring.kafka.producer.batch-size= *# Number of records to batch before sending.*

spring.kafka.producer.bootstrap-servers= *# Comma-delimited list of host:port pairs to use for establishing the initial connection to the Kafka cluster.*

spring.kafka.producer.buffer-memory= *# Total bytes of memory the producer can use to buffer records waiting to be sent to the server.*

spring.kafka.producer.client-id= *# Id to pass to the server when making requests; used for server-side logging.*

spring.kafka.producer.compression-type= *# Compression type for all data generated by the producer.*

spring.kafka.producer.key-serializer= *# Serializer class for keys.*

spring.kafka.producer.retries= *# When greater than zero, enables retrying of failed sends.*

spring.kafka.producer.value-serializer= *# Serializer class for values.*

spring.kafka.properties.\*= *# Additional properties used to configure the client.*

spring.kafka.ssl.key-password= *# Password of the private key in the key store file.*

spring.kafka.ssl.keystore-location= *# Location of the key store file.*

spring.kafka.ssl.keystore-password= *# Store password for the key store file.*

spring.kafka.ssl.truststore-location= *# Location of the trust store file.*

spring.kafka.ssl.truststore-password= *# Store password for the trust store file.*

spring.kafka.template.default-topic= *# Default topic to which messages will be sent.*

*# RABBIT (*[RabbitProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/amqp/RabbitProperties.java))

spring.rabbitmq.addresses= *# Comma-separated list of addresses to which the client should connect.*

spring.rabbitmq.cache.channel.checkout-timeout= *# Number of milliseconds to wait to obtain a channel if the cache size has been reached.*

spring.rabbitmq.cache.channel.size= *# Number of channels to retain in the cache.*

spring.rabbitmq.cache.connection.mode=channel *# Connection factory cache mode.*

spring.rabbitmq.cache.connection.size= *# Number of connections to cache.*

spring.rabbitmq.connection-timeout= *# Connection timeout, in milliseconds; zero for infinite.*

spring.rabbitmq.dynamic=true *# Create an AmqpAdmin bean.*

spring.rabbitmq.host=localhost *# RabbitMQ host.*

spring.rabbitmq.listener.acknowledge-mode= *# Acknowledge mode of container.*

spring.rabbitmq.listener.auto-startup=true *# Start the container automatically on startup.*

spring.rabbitmq.listener.concurrency= *# Minimum number of consumers.*

spring.rabbitmq.listener.default-requeue-rejected= *# Whether or not to requeue delivery failures; default `true`.*

spring.rabbitmq.listener.idle-event-interval= *# How often idle container events should be published in milliseconds.*

spring.rabbitmq.listener.max-concurrency= *# Maximum number of consumers.*

spring.rabbitmq.listener.prefetch= *# Number of messages to be handled in a single request. It should be greater than or equal to the transaction size (if used).*

spring.rabbitmq.listener.retry.enabled=false *# Whether or not publishing retries are enabled.*

spring.rabbitmq.listener.retry.initial-interval=1000 *# Interval between the first and second attempt to deliver a message.*

spring.rabbitmq.listener.retry.max-attempts=3 *# Maximum number of attempts to deliver a message.*

spring.rabbitmq.listener.retry.max-interval=10000 *# Maximum interval between attempts.*

spring.rabbitmq.listener.retry.multiplier=1.0 *# A multiplier to apply to the previous delivery retry interval.*

spring.rabbitmq.listener.retry.stateless=true *# Whether or not retry is stateless or stateful.*

spring.rabbitmq.listener.transaction-size= *# Number of messages to be processed in a transaction. For best results it should be less than or equal to the prefetch count.*

spring.rabbitmq.password= *# Login to authenticate against the broker.*

spring.rabbitmq.port=5672 *# RabbitMQ port.*

spring.rabbitmq.publisher-confirms=false *# Enable publisher confirms.*

spring.rabbitmq.publisher-returns=false *# Enable publisher returns.*

spring.rabbitmq.requested-heartbeat= *# Requested heartbeat timeout, in seconds; zero for none.*

spring.rabbitmq.ssl.enabled=false *# Enable SSL support.*

spring.rabbitmq.ssl.key-store= *# Path to the key store that holds the SSL certificate.*

spring.rabbitmq.ssl.key-store-password= *# Password used to access the key store.*

spring.rabbitmq.ssl.trust-store= *# Trust store that holds SSL certificates.*

spring.rabbitmq.ssl.trust-store-password= *# Password used to access the trust store.*

spring.rabbitmq.ssl.algorithm= *# SSL algorithm to use. By default configure by the rabbit client library.*

spring.rabbitmq.template.mandatory=false *# Enable mandatory messages.*

spring.rabbitmq.template.receive-timeout=0 *# Timeout for `receive()` methods.*

spring.rabbitmq.template.reply-timeout=5000 *# Timeout for `sendAndReceive()` methods.*

spring.rabbitmq.template.retry.enabled=false *# Set to true to enable retries in the `RabbitTemplate`.*

spring.rabbitmq.template.retry.initial-interval=1000 *# Interval between the first and second attempt to publish a message.*

spring.rabbitmq.template.retry.max-attempts=3 *# Maximum number of attempts to publish a message.*

spring.rabbitmq.template.retry.max-interval=10000 *# Maximum number of attempts to publish a message.*

spring.rabbitmq.template.retry.multiplier=1.0 *# A multiplier to apply to the previous publishing retry interval.*

spring.rabbitmq.username= *# Login user to authenticate to the broker.*

spring.rabbitmq.virtual-host= *# Virtual host to use when connecting to the broker.*

*# ----------------------------------------*

*# ACTUATOR PROPERTIES*

*# ----------------------------------------*

*# ENDPOINTS (*[AbstractEndpoint](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-actuator/src/main/java/org/springframework/boot/actuate/endpoint/AbstractEndpoint.java) subclasses)

endpoints.enabled=true *# Enable endpoints.*

endpoints.sensitive= *# Default endpoint sensitive setting.*

endpoints.actuator.enabled=true *# Enable the endpoint.*

endpoints.actuator.path= *# Endpoint URL path.*

endpoints.actuator.sensitive=false *# Enable security on the endpoint.*

endpoints.auditevents.enabled= *# Enable the endpoint.*

endpoints.auditevents.path= *# Endpoint path.*

endpoints.auditevents.sensitive=false *# Enable security on the endpoint.*

endpoints.autoconfig.enabled= *# Enable the endpoint.*

endpoints.autoconfig.id= *# Endpoint identifier.*

endpoints.autoconfig.path= *# Endpoint path.*

endpoints.autoconfig.sensitive= *# Mark if the endpoint exposes sensitive information.*

endpoints.beans.enabled= *# Enable the endpoint.*

endpoints.beans.id= *# Endpoint identifier.*

endpoints.beans.path= *# Endpoint path.*

endpoints.beans.sensitive= *# Mark if the endpoint exposes sensitive information.*

endpoints.configprops.enabled= *# Enable the endpoint.*

endpoints.configprops.id= *# Endpoint identifier.*

endpoints.configprops.keys-to-sanitize=password,secret,key,token,.\*credentials.\*,vcap\_services *# Keys that should be sanitized. Keys can be simple strings that the property ends with or regex expressions.*

endpoints.configprops.path= *# Endpoint path.*

endpoints.configprops.sensitive= *# Mark if the endpoint exposes sensitive information.*

endpoints.docs.curies.enabled=false *# Enable the curie generation.*

endpoints.docs.enabled=true *# Enable actuator docs endpoint.*

endpoints.docs.path=/docs *#*

endpoints.docs.sensitive=false *#*

endpoints.dump.enabled= *# Enable the endpoint.*

endpoints.dump.id= *# Endpoint identifier.*

endpoints.dump.path= *# Endpoint path.*

endpoints.dump.sensitive= *# Mark if the endpoint exposes sensitive information.*

endpoints.env.enabled= *# Enable the endpoint.*

endpoints.env.id= *# Endpoint identifier.*

endpoints.env.keys-to-sanitize=password,secret,key,token,.\*credentials.\*,vcap\_services *# Keys that should be sanitized. Keys can be simple strings that the property ends with or regex expressions.*

endpoints.env.path= *# Endpoint path.*

endpoints.env.sensitive= *# Mark if the endpoint exposes sensitive information.*

endpoints.flyway.enabled= *# Enable the endpoint.*

endpoints.flyway.id= *# Endpoint identifier.*

endpoints.flyway.sensitive= *# Mark if the endpoint exposes sensitive information.*

endpoints.health.enabled= *# Enable the endpoint.*

endpoints.health.id= *# Endpoint identifier.*

endpoints.health.mapping.\*= *# Mapping of health statuses to HttpStatus codes. By default, registered health statuses map to sensible defaults (i.e. UP maps to 200).*

endpoints.health.path= *# Endpoint path.*

endpoints.health.sensitive= *# Mark if the endpoint exposes sensitive information.*

endpoints.health.time-to-live=1000 *# Time to live for cached result, in milliseconds.*

endpoints.heapdump.enabled= *# Enable the endpoint.*

endpoints.heapdump.path= *# Endpoint path.*

endpoints.heapdump.sensitive= *# Mark if the endpoint exposes sensitive information.*

endpoints.hypermedia.enabled=false *# Enable hypermedia support for endpoints.*

endpoints.info.enabled= *# Enable the endpoint.*

endpoints.info.id= *# Endpoint identifier.*

endpoints.info.path= *# Endpoint path.*

endpoints.info.sensitive= *# Mark if the endpoint exposes sensitive information.*

endpoints.jolokia.enabled=true *# Enable Jolokia endpoint.*

endpoints.jolokia.path=/jolokia *# Endpoint URL path.*

endpoints.jolokia.sensitive=true *# Enable security on the endpoint.*

endpoints.liquibase.enabled= *# Enable the endpoint.*

endpoints.liquibase.id= *# Endpoint identifier.*

endpoints.liquibase.sensitive= *# Mark if the endpoint exposes sensitive information.*

endpoints.logfile.enabled=true *# Enable the endpoint.*

endpoints.logfile.external-file= *# External Logfile to be accessed.*

endpoints.logfile.path=/logfile *# Endpoint URL path.*

endpoints.logfile.sensitive=true *# Enable security on the endpoint.*

endpoints.loggers.enabled=true *# Enable the endpoint.*

endpoints.loggers.id= *# Endpoint identifier.*

endpoints.loggers.path=/logfile *# Endpoint path.*

endpoints.loggers.sensitive=true *# Mark if the endpoint exposes sensitive information.*

endpoints.mappings.enabled= *# Enable the endpoint.*

endpoints.mappings.id= *# Endpoint identifier.*

endpoints.mappings.path= *# Endpoint path.*

endpoints.mappings.sensitive= *# Mark if the endpoint exposes sensitive information.*

endpoints.metrics.enabled= *# Enable the endpoint.*

endpoints.metrics.filter.enabled=true *# Enable the metrics servlet filter.*

endpoints.metrics.filter.gauge-submissions=merged *# Http filter gauge submissions (merged, per-http-method)*

endpoints.metrics.filter.counter-submissions=merged *# Http filter counter submissions (merged, per-http-method)*

endpoints.metrics.id= *# Endpoint identifier.*

endpoints.metrics.path= *# Endpoint path.*

endpoints.metrics.sensitive= *# Mark if the endpoint exposes sensitive information.*

endpoints.shutdown.enabled= *# Enable the endpoint.*

endpoints.shutdown.id= *# Endpoint identifier.*

endpoints.shutdown.path= *# Endpoint path.*

endpoints.shutdown.sensitive= *# Mark if the endpoint exposes sensitive information.*

endpoints.trace.enabled= *# Enable the endpoint.*

endpoints.trace.id= *# Endpoint identifier.*

endpoints.trace.path= *# Endpoint path.*

endpoints.trace.sensitive= *# Mark if the endpoint exposes sensitive information.*

*# ENDPOINTS CORS CONFIGURATION (*[EndpointCorsProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-actuator/src/main/java/org/springframework/boot/actuate/autoconfigure/EndpointCorsProperties.java))

endpoints.cors.allow-credentials= *# Set whether credentials are supported. When not set, credentials are not supported.*

endpoints.cors.allowed-headers= *# Comma-separated list of headers to allow in a request. '\*' allows all headers.*

endpoints.cors.allowed-methods=GET *# Comma-separated list of methods to allow. '\*' allows all methods.*

endpoints.cors.allowed-origins= *# Comma-separated list of origins to allow. '\*' allows all origins. When not set, CORS support is disabled.*

endpoints.cors.exposed-headers= *# Comma-separated list of headers to include in a response.*

endpoints.cors.max-age=1800 *# How long, in seconds, the response from a pre-flight request can be cached by clients.*

*# JMX ENDPOINT (*[EndpointMBeanExportProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-actuator/src/main/java/org/springframework/boot/actuate/autoconfigure/EndpointMBeanExportProperties.java))

endpoints.jmx.domain= *# JMX domain name. Initialized with the value of 'spring.jmx.default-domain' if set.*

endpoints.jmx.enabled=true *# Enable JMX export of all endpoints.*

endpoints.jmx.static-names= *# Additional static properties to append to all ObjectNames of MBeans representing Endpoints.*

endpoints.jmx.unique-names=false *# Ensure that ObjectNames are modified in case of conflict.*

*# JOLOKIA (*[JolokiaProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-actuator/src/main/java/org/springframework/boot/actuate/autoconfigure/JolokiaProperties.java))

jolokia.config.\*= *# See Jolokia manual*

*# MANAGEMENT HTTP SERVER (*[ManagementServerProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-actuator/src/main/java/org/springframework/boot/actuate/autoconfigure/ManagementServerProperties.java))

management.add-application-context-header=true *# Add the "X-Application-Context" HTTP header in each response.*

management.address= *# Network address that the management endpoints should bind to.*

management.context-path= *# Management endpoint context-path. For instance `/actuator`*

management.cloudfoundry.enabled= *# Enable extended Cloud Foundry actuator endpoints*

management.cloudfoundry.skip-ssl-validation= *# Skip SSL verification for Cloud Foundry actuator endpoint security calls*

management.port= *# Management endpoint HTTP port. Uses the same port as the application by default. Configure a different port to use management-specific SSL.*

management.security.enabled=true *# Enable security.*

management.security.roles=ACTUATOR *# Comma-separated list of roles that can access the management endpoint.*

management.security.sessions=stateless *# Session creating policy to use (always, never, if\_required, stateless).*

management.ssl.ciphers= *# Supported SSL ciphers. Requires a custom management.port.*

management.ssl.client-auth= *# Whether client authentication is wanted ("want") or needed ("need"). Requires a trust store. Requires a custom management.port.*

management.ssl.enabled= *# Enable SSL support. Requires a custom management.port.*

management.ssl.enabled-protocols= *# Enabled SSL protocols. Requires a custom management.port.*

management.ssl.key-alias= *# Alias that identifies the key in the key store. Requires a custom management.port.*

management.ssl.key-password= *# Password used to access the key in the key store. Requires a custom management.port.*

management.ssl.key-store= *# Path to the key store that holds the SSL certificate (typically a jks file). Requires a custom management.port.*

management.ssl.key-store-password= *# Password used to access the key store. Requires a custom management.port.*

management.ssl.key-store-provider= *# Provider for the key store. Requires a custom management.port.*

management.ssl.key-store-type= *# Type of the key store. Requires a custom management.port.*

management.ssl.protocol=TLS *# SSL protocol to use. Requires a custom management.port.*

management.ssl.trust-store= *# Trust store that holds SSL certificates. Requires a custom management.port.*

management.ssl.trust-store-password= *# Password used to access the trust store. Requires a custom management.port.*

management.ssl.trust-store-provider= *# Provider for the trust store. Requires a custom management.port.*

management.ssl.trust-store-type= *# Type of the trust store. Requires a custom management.port.*

*# HEALTH INDICATORS*

management.health.db.enabled=true *# Enable database health check.*

management.health.cassandra.enabled=true *# Enable cassandra health check.*

management.health.couchbase.enabled=true *# Enable couchbase health check.*

management.health.defaults.enabled=true *# Enable default health indicators.*

management.health.diskspace.enabled=true *# Enable disk space health check.*

management.health.diskspace.path= *# Path used to compute the available disk space.*

management.health.diskspace.threshold=0 *# Minimum disk space that should be available, in bytes.*

management.health.elasticsearch.enabled=true *# Enable elasticsearch health check.*

management.health.elasticsearch.indices= *# Comma-separated index names.*

management.health.elasticsearch.response-timeout=100 *# The time, in milliseconds, to wait for a response from the cluster.*

management.health.jms.enabled=true *# Enable JMS health check.*

management.health.ldap.enabled=true *# Enable LDAP health check.*

management.health.mail.enabled=true *# Enable Mail health check.*

management.health.mongo.enabled=true *# Enable MongoDB health check.*

management.health.rabbit.enabled=true *# Enable RabbitMQ health check.*

management.health.redis.enabled=true *# Enable Redis health check.*

management.health.solr.enabled=true *# Enable Solr health check.*

management.health.status.order=DOWN, OUT\_OF\_SERVICE, UP, UNKNOWN *# Comma-separated list of health statuses in order of severity.*

*# INFO CONTRIBUTORS (*[InfoContributorProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-actuator/src/main/java/org/springframework/boot/actuate/autoconfigure/InfoContributorProperties.java))

management.info.build.enabled=true *# Enable build info.*

management.info.defaults.enabled=true *# Enable default info contributors.*

management.info.env.enabled=true *# Enable environment info.*

management.info.git.enabled=true *# Enable git info.*

management.info.git.mode=simple *# Mode to use to expose git information.*

*# REMOTE SHELL (*[ShellProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-actuator/src/main/java/org/springframework/boot/actuate/autoconfigure/ShellProperties.java))

management.shell.auth.type=simple *# Authentication type. Auto-detected according to the environment.*

management.shell.auth.jaas.domain=my-domain *# JAAS domain.*

management.shell.auth.key.path= *# Path to the authentication key. This should point to a valid ".pem" file.*

management.shell.auth.simple.user.name=user *# Login user.*

management.shell.auth.simple.user.password= *# Login password.*

management.shell.auth.spring.roles=ACTUATOR *# Comma-separated list of required roles to login to the CRaSH console.*

management.shell.command-path-patterns=classpath\*:/commands/\*\*,classpath\*:/crash/commands/\*\* *# Patterns to use to look for commands.*

management.shell.command-refresh-interval=-1 *# Scan for changes and update the command if necessary (in seconds).*

management.shell.config-path-patterns=classpath\*:/crash/\* *# Patterns to use to look for configurations.*

management.shell.disabled-commands=jpa\*,jdbc\*,jndi\* *# Comma-separated list of commands to disable.*

management.shell.disabled-plugins= *# Comma-separated list of plugins to disable. Certain plugins are disabled by default based on the environment.*

management.shell.ssh.auth-timeout = *# Number of milliseconds after user will be prompted to login again.*

management.shell.ssh.enabled=true *# Enable CRaSH SSH support.*

management.shell.ssh.idle-timeout = *# Number of milliseconds after which unused connections are closed.*

management.shell.ssh.key-path= *# Path to the SSH server key.*

management.shell.ssh.port=2000 *# SSH port.*

management.shell.telnet.enabled=false *# Enable CRaSH telnet support. Enabled by default if the TelnetPlugin is available.*

management.shell.telnet.port=5000 *# Telnet port.*

*# TRACING (*[TraceProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-actuator/src/main/java/org/springframework/boot/actuate/trace/TraceProperties.java))

management.trace.include=request-headers,response-headers,cookies,errors *# Items to be included in the trace.*

*# METRICS EXPORT (*[MetricExportProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-actuator/src/main/java/org/springframework/boot/actuate/metrics/export/MetricExportProperties.java))

spring.metrics.export.aggregate.key-pattern= *# Pattern that tells the aggregator what to do with the keys from the source repository.*

spring.metrics.export.aggregate.prefix= *# Prefix for global repository if active.*

spring.metrics.export.delay-millis=5000 *# Delay in milliseconds between export ticks. Metrics are exported to external sources on a schedule with this delay.*

spring.metrics.export.enabled=true *# Flag to enable metric export (assuming a MetricWriter is available).*

spring.metrics.export.excludes= *# List of patterns for metric names to exclude. Applied after the includes.*

spring.metrics.export.includes= *# List of patterns for metric names to include.*

spring.metrics.export.redis.key=keys.spring.metrics *# Key for redis repository export (if active).*

spring.metrics.export.redis.prefix=spring.metrics *# Prefix for redis repository if active.*

spring.metrics.export.send-latest= *# Flag to switch off any available optimizations based on not exporting unchanged metric values.*

spring.metrics.export.statsd.host= *# Host of a statsd server to receive exported metrics.*

spring.metrics.export.statsd.port=8125 *# Port of a statsd server to receive exported metrics.*

spring.metrics.export.statsd.prefix= *# Prefix for statsd exported metrics.*

spring.metrics.export.triggers.\*= *# Specific trigger properties per MetricWriter bean name.*

*# ----------------------------------------*

*# DEVTOOLS PROPERTIES*

*# ----------------------------------------*

*# DEVTOOLS (*[DevToolsProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-devtools/src/main/java/org/springframework/boot/devtools/autoconfigure/DevToolsProperties.java))

spring.devtools.livereload.enabled=true *# Enable a livereload.com compatible server.*

spring.devtools.livereload.port=35729 *# Server port.*

spring.devtools.restart.additional-exclude= *# Additional patterns that should be excluded from triggering a full restart.*

spring.devtools.restart.additional-paths= *# Additional paths to watch for changes.*

spring.devtools.restart.enabled=true *# Enable automatic restart.*

spring.devtools.restart.exclude=META-INF/maven/\*\*,META-INF/resources/\*\*,resources/\*\*,static/\*\*,public/\*\*,templates/\*\*,\*\*/\*Test.class,\*\*/\*Tests.class,git.properties *# Patterns that should be excluded from triggering a full restart.*

spring.devtools.restart.poll-interval=1000 *# Amount of time (in milliseconds) to wait between polling for classpath changes.*

spring.devtools.restart.quiet-period=400 *# Amount of quiet time (in milliseconds) required without any classpath changes before a restart is triggered.*

spring.devtools.restart.trigger-file= *# Name of a specific file that when changed will trigger the restart check. If not specified any classpath file change will trigger the restart.*

*# REMOTE DEVTOOLS (*[RemoteDevToolsProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.2.RELEASE/spring-boot-devtools/src/main/java/org/springframework/boot/devtools/autoconfigure/RemoteDevToolsProperties.java))

spring.devtools.remote.context-path=/.~~spring-boot!~ *# Context path used to handle the remote connection.*

spring.devtools.remote.debug.enabled=true *# Enable remote debug support.*

spring.devtools.remote.debug.local-port=8000 *# Local remote debug server port.*

spring.devtools.remote.proxy.host= *# The host of the proxy to use to connect to the remote application.*

spring.devtools.remote.proxy.port= *# The port of the proxy to use to connect to the remote application.*

spring.devtools.remote.restart.enabled=true *# Enable remote restart.*

spring.devtools.remote.secret= *# A shared secret required to establish a connection (required to enable remote support).*

spring.devtools.remote.secret-header-name=X-AUTH-TOKEN *# HTTP header used to transfer the shared secret.*

*# ----------------------------------------*

*# TESTING PROPERTIES*

*# ----------------------------------------*

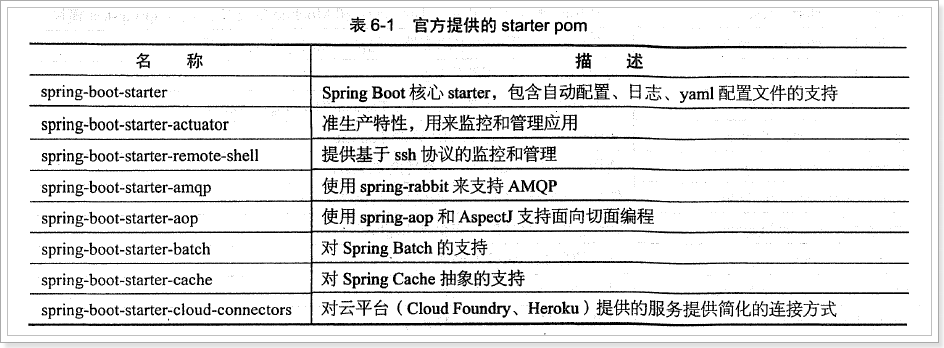
spring.test.database.replace=any *# Type of existing DataSource to replace.*

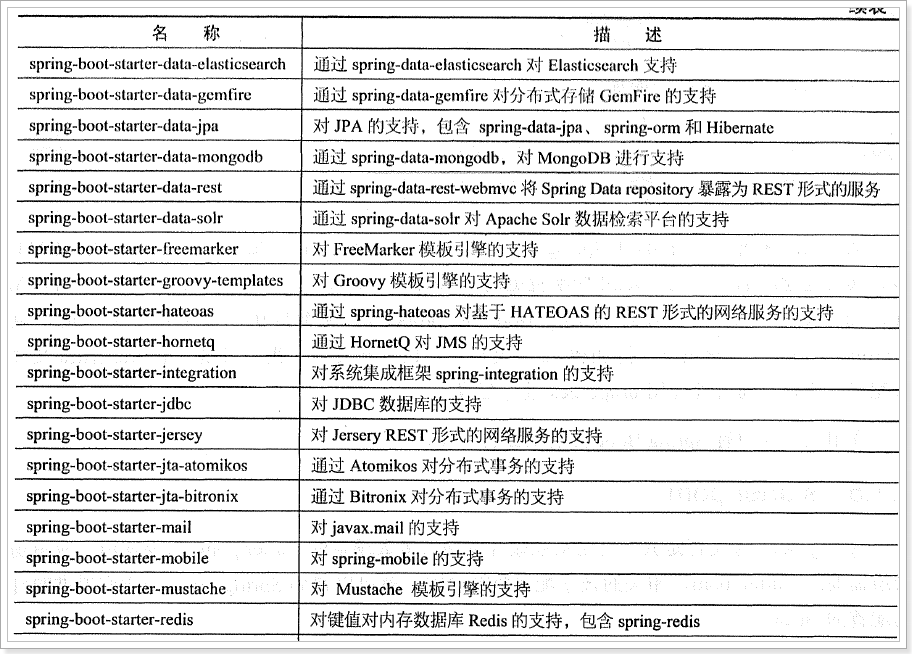
spring.test.mockmvc.print=default *# MVC Print option.*

8.**starter pom:**

SpringBoot为我们提供了企业级开发绝大多数场景的starter pom,只要使用了应用场景所需的starter pom，相关的技术配置将会消除，就可以得到Spring

Boot为我们提供的自动配置的Bean。





****

1. **读取自定义配置：**

(1)在application.properties文件中自定义属性：例如

book.author=张三

book.name=SpringBoot

(2)使用@Value(“${book.author}”) 获取自定义属性的值

1. **类型安全的配置：**

使用@Value注入每个自定义配置在项目中显得很麻烦，当自定义属性很多时需要注入很多次

SpringBoot还提供了基于类型安全的配置方式，通过@ConfigurationProperties将properties中的属性和一个Bean的属性关联，从而实现类型安全的配置

(1)在application.properties文件中自定义属性：例如

book.author=张三

book.name=SpringBoot

1. @ConfigurationProperties(prefix=”book”)
2. **Profile配置:**

Profile是针对不同的环境对不同的配置提供支持的，全局Profile配置使用application-\*.properties

(application-prod.properties,application-sit.properties,application-dev.properties)

通过在application.properties中设置spring.profiles.active=dev来指定活动的Profile.

1. **SpringBoot自动配置的原理**
2. **创建父工程：**

<dependencyManagement>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-dependencies</artifactId>

<version>1.5.6.RELEASE</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

</plugins>

</build>

1. **SpringBoot整合测试:**

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<scope>test</scope>

</dependency>

1. **15.@SpringBootApplication注解**

该注解是一个组合注解

**16.SpringBoot的日志管理：**

SprongBoot使用的默认日志框架是Logback,并用INFO级别输出到控制台:

日志输出内容元素具体如下：

* 时间日期：精确到毫秒
* 日志级别：ERROR, WARN, INFO, DEBUG or TRACE
* 进程ID
* 分隔符：— 标识实际日志的开始
* 线程名：方括号括起来（可能会截断控制台输出)
* Logger名：通常使用源代码的类名
* 日志内容

日志依赖:该依赖内容就是 Spring Boot 默认的日志框架 logback

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-logging</artifactId>

</dependency>

实际开发中我们不需要直接添加该依赖。

SpringBoot的日志的级别有7个：  
TRACE , DEBUG , INFO , WARN , ERROR , FATAL , OFF

日志级别从低到高为:

TRACE < DEBUG < INFO < WARN < ERROR < FATAL< OFF 。

如果设置为 WARN ，则低于 WARN 的信息都不会输出。

Spring Boot 中默认配置 ERROR 、 WARN 和 INFO 级别的日志输出到控制台。

例如

#root日志以WARN级别输出

logging.level.root=WARN (让日志只输出warn及以上级别的信息)

#springframework.web日志以DEBUG级别输出

logging.level.org.springframework.web=DEBUG

#hibernate日志以ERROR级别输出

logging.level.org.hibernate=ERROR

默认情况下，Spring Boot将日志输出到控制台，不会写到日志文件。如果要编写除控制台输出之外的日志文件，则需在application.properties中设置logging.file或logging.path属性。

logging.file，设置文件，可以是绝对路径，也可以是相对路径。如：logging.file=log/my.log(相对)或者/log/my.log(绝对)

* logging.path，设置目录，会在该目录下创建spring.log文件，并写入日志内容，如：logging.path=/var/log

如果只配置 logging.file，会在项目的当前路径下生成一个 xxx.log 日志文件。  
如果只配置 logging.path，在 /var/log文件夹生成一个日志文件为 spring.log

**注：二者不能同时使用，如若同时使用，则只有logging.file生效**

默认情况下，日志文件的大小达到10MB时会切分一次，产生新的日志文件，默认级别为：ERROR、WARN、INFO

#配置日志

logging.level.root=WARN

logging.level.org.springframework.web=DEBUG

logging.file=e:\\springboot\\info.log

logging.pattern.console=%d{yyyy/MM/dd-HH:mm:ss} [%thread] %-5level %logger- %msg%n

logging.pattern.file=%d{yyyy/MM/dd-HH:mm} [%thread] %-5level %logger- %msg%n

## ****自定义日志配置****

通过系统属性和传统的Spring Boot外部配置文件依然可以很好的支持日志控制和管理。

根据不同的日志系统，你可以按如下规则组织配置文件名，就能被正确加载：

* Logback：logback-spring.xml, logback-spring.groovy, logback.xml, logback.groovy
* Log4j：log4j-spring.properties, log4j-spring.xml, log4j.properties, log4j.xml
* Log4j2：log4j2-spring.xml, log4j2.xml
* JDK (Java Util Logging)：logging.properties

Spring Boot官方推荐优先使用带有-spring的文件名作为你的日志配置（如使用logback-spring.xml，而不是logback.xml），命名为logback-spring.xml的日志配置文件，spring boot可以为它添加一些spring boot特有的配置项。

如果你即想完全掌控日志配置，但又不想用logback.xml作为Logback配置的名字，可以通过logging.config属性指定自定义的名字：

logging.config=classpath:logging-config.xml

虽然一般并不需要改变配置文件的名字，但是如果你想针对不同运行时Profile使用不同的日志配置，这个功能会很有用。

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<configuration scan=*"true"* scanPeriod=*"60 seconds"* debug=*"false"*>

<contextName>logback</contextName>

<property name=*"log.path"* value=*"E:\\springboot\\info.log"* />

<!--输出到控制台-->

<appender name=*"console"* class=*"ch.qos.logback.core.ConsoleAppender"*>

<!-- <filter class="ch.qos.logback.classic.filter.ThresholdFilter">

<level>ERROR</level>

</filter>-->

<encoder>

<pattern>%d{HH:mm:ss.SSS} %contextName [%thread] %-5level %logger{36} - %msg%n</pattern>

</encoder>

</appender>

<!--输出到文件-->

<appender name=*"file"* class=*"ch.qos.logback.core.rolling.RollingFileAppender"*>

<file>${log.path}</file>

<rollingPolicy class=*"ch.qos.logback.core.rolling.TimeBasedRollingPolicy"*>

<fileNamePattern>logback.%d{yyyy-MM-dd}.log</fileNamePattern>

</rollingPolicy>

<encoder>

<pattern>%d{HH:mm:ss.SSS} %contextName [%thread] %-5level %logger{36} - %msg%n</pattern>

</encoder>

</appender>

<root level=*"info"*>

<appender-ref ref=*"console"* />

<appender-ref ref=*"file"* />

</root>

<!-- logback为java中的包 -->

<logger name=*"com.borun.controller"*/>

<!--additivity是否向上级loger传递打印信息-->

<logger name=*"com.borun.controller.SpringController"* level=*"WARN"* additivity=*"false"*>

<appender-ref ref=*"console"*/>

</logger>

</configuration>

#### 根节点<configuration>包含的属性:

* scan:当此属性设置为true时，配置文件如果发生改变，将会被重新加载，默认值为true。
* scanPeriod:设置监测配置文件是否有修改的时间间隔，如果没有给出时间单位，默认单位是毫秒。当scan为true时，此属性生效。默认的时间间隔为1分钟。
* debug:当此属性设置为true时，将打印出logback内部日志信息，实时查看logback运行状态。默认值为false。

<contextname>:每个logger都关联到logger上下文，默认上下文名称为“default”。但可以使用设置成其他名字，用于区分不同应用程序的记录。一旦设置，不能修改,可以通过%contextName来打印日志上下文名称。

<property>:用来定义变量值的标签， 有两个属性，name和value；其中name的值是变量的名称，value的值时变量定义的值。通过定义的值会被插入到logger上下文中。定义变量后，可以使“${}”来使用变量。

<appender>:用来格式化日志输出节点，有俩个属性name和class，class用来指定哪种输出策略，常用就是控制台输出策略和文件输出策略。

ThresholdFilter为系统定义的拦截器，例如我们用ThresholdFilter来过滤掉ERROR级别以下的日志不输出到文件中。如果不用记得注释掉，不然你控制台会发现没日志。

RollingFileAppender用于切分文件日志：

<fileNamePattern>logback.%d{yyyy-MM-dd}.log</fileNamePattern>:定义了日志的切分方式——把每一天的日志归档到一个文件中，

<maxHistory>30</maxHistory>:表示只保留最近30天的日志，以防止日志填满整个磁盘空间。同理，可以使用%d{yyyy-MM-dd\_HH-mm}来定义精确到分的日志切分方式。

<totalSizeCap>1GB</totalSizeCap>用来指定日志文件的上限大小，例如设置为1GB的话，那么到了这个值，就会删除旧的日志。

<root>节点是必选节点，用来指定最基础的日志输出级别，只有一个level属性。

<loger>用来设置某一个包或者具体的某一个类的日志打印级别、以及指定<appender>。<loger>仅有一个name属性，一个可选的level和一个可选的addtivity属性。

name:用来指定受此loger约束的某一个包或者具体的某一个类。

level:用来设置打印级别，大小写无关：TRACE, DEBUG, INFO, WARN, ERROR, ALL 和 OFF，还有一个特俗值INHERITED或者同义词NULL，代表强制执行上级的级别。如果未设置此属性，那么当前loger将会继承上级的级别。

* addtivity:是否向上级loger传递打印信息。默认是true。

1. **loger的使用：**

第一种：带有loger的配置，不指定级别，不指定appender

<logger name=*"com.borun.controller"*/>

将控制controller包下的所有类的日志的打印，但是并没用设置打印级别，所以继承他的上级的日志级别“info”；  
没有设置addtivity，默认为true，将此loger的打印信息向上级传递；  
没有设置appender，此loger本身不打印任何信息。

当执行该包下的某个类中的功能时，将级别为“info”及大于“info”的日志信息传递给root，本身并不打印；  
root接到下级传递的信息，交给已经配置好的名为“console”的appender处理，“console”appender将信息打印到控制台。

第二种：带有多个loger的配置，指定级别，指定appender

<logger name=*"com.borun.controller.SpringController"* level=*"WARN"* additivity=*"false"*>

<appender-ref ref=*"console"*/>

</logger>

控制com.borun.controller.SpringController类的日志打印，打印级别为“WARN”;  
additivity属性为false，表示此loger的打印信息不再向上级传递;  
指定了名字为“console”的appender;

当执行SpringController中的方法时将级别为“WARN”及大于“WARN”的日志信息交给此loger指定的名为“console”的appender处理，在控制台中打出日志，不再向上级root传递打印信息。

### 多环境日志输出:

根据不同环境（prod:生产环境，test:测试环境，dev:开发环境）来定义不同的日志输出，在 logback-spring.xml中使用 springProfile 节点来定义，方法如下：

<!-- 测试环境+开发环境. 多个使用逗号隔开-->

<springProfile name=*"test,dev"*>

<logger name=*"com.borun.controller"* level=*"info"* />

</springProfile>

<!-- 生产环境-->

<springProfile name=*"prod"*>

<logger name=*"com.borun.controller"* level=*"ERROR"* />

</springProfile>

在application.properties文件中指明使用哪一种：

spring.profiles.active=prod

**14.使用log4j日志管理:**

1.修改pom.xml文件，过滤掉自带的spring-boot-starter-logging，然后添加spring-boot-starter-log4j依赖包。

<!-- log4j -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter</artifactId>

<exclusions>

<exclusion>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-logging</artifactId>

</exclusion>

</exclusions>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-log4j</artifactId>

<version>1.3.8.RELEASE</version>

</dependency>

2.在resources目录下新建log4j.properties配置文件

 配置info,error,console,debug四种输出格式。

[IMG_256](https://www.cnblogs.com/30go/p/javascript:void(0);)

log4j.rootLogger=info,error,CONSOLE,DEBUG

log4j.appender.CONSOLE=org.apache.log4j.ConsoleAppender

log4j.appender.CONSOLE.layout=org.apache.log4j.PatternLayout

log4j.appender.CONSOLE.layout.ConversionPattern=%d{yyyy-MM-dd-HH-mm} [%t] [%c] [%p] - %m%n

log4j.logger.info=info

log4j.appender.info=org.apache.log4j.DailyRollingFileAppender

log4j.appender.info.layout=org.apache.log4j.PatternLayout

log4j.appender.info.layout.ConversionPattern=%d{yyyy-MM-dd-HH-mm} [%t] [%c] [%p] - %m%n

log4j.appender.info.datePattern='.'yyyy-MM-dd

log4j.appender.info.Threshold = info

log4j.appender.info.append=true

log4j.appender.info.File=e://springboot/api\_services\_info.log

log4j.logger.error=error

log4j.appender.error=org.apache.log4j.DailyRollingFileAppender

log4j.appender.error.layout=org.apache.log4j.PatternLayout

log4j.appender.error.layout.ConversionPattern=%d{yyyy-MM-dd-HH-mm} [%t] [%c] [%p] - %m%n

log4j.appender.error.datePattern='.'yyyy-MM-dd

log4j.appender.error.Threshold = error

log4j.appender.error.append=true

log4j.appender.error.File=e://springboot/api\_services\_error.log

log4j.logger.DEBUG=DEBUG

log4j.appender.DEBUG=org.apache.log4j.DailyRollingFileAppender

log4j.appender.DEBUG.layout=org.apache.log4j.PatternLayout

log4j.appender.DEBUG.layout.ConversionPattern=%d{yyyy-MM-dd-HH-mm} [%t] [%c] [%p] - %m%n

log4j.appender.DEBUG.datePattern='.'yyyy-MM-dd

log4j.appender.DEBUG.Threshold = DEBUG

log4j.appender.DEBUG.append=true

log4j.appender.DEBUG.File=e://springboot/api\_services\_debug.log

1. 在使用到的类中声明log以及输出log信息。

@RestController

public class UserController{

private static Logger log = Logger.getLogger(UserController.class);

@Autowired

UserService userService;

@RequestMapping(“/add”)

public String addUser(){

User u=new User(“zhangsan”，“12345”);

int result = userService.addUser(u);

log.info(“增加用户成功”+result);

return “success”+result;

}

}

**15.配置为开发模式：**

<!-- 加入以下依赖，代码做了修改，不用重新运行 -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>springloaded</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId> </dependency>

**16.SpringBoot的Web开发:**

SpringBoot提供了spring-boot-starter-web为Web开发予以支持，spring-boot-starter-web提供了嵌入的Tomcat以及SpringMvc的依赖，Web相关的自动配置存储在spring-boot-autoconfigure.jar的org.springframework.boot.autoconfigure.web下。

**17.访问静态资源:**

在SpringBoot中加载静态资源和在普通的web应用中不太一样。默认情况下，Spring Boot从classpath的/static,/public或/META-INF/resources文件夹或从ServletContext根目录提供静态内容

#设定静态文件路径，js,css,image等spring.resources.static-locations=classpath:/static/

1. **自定义消息转化器**

只需要在类中添加消息转化器的@Bean，就会被Spring Boot自动加入到容器中。

@Bean

**public** StringHttpMessageConverter stringHttpMessageConverter(){

StringHttpMessageConverter converter=**new** StringHttpMessageConverter(Charset.*forName*("UTF-8"));

**return** converter;}

**19.自定义拦截器:**

有些时候我们需要自已配置SpringMVC而不是采用默认，比如说增加一个拦截器，这个时候就得通过继承WebMvcConfigurerAdapter然后重写父类中的方法进行扩展。

@Configuration

**public** **class** MyInterceptor **extends** WebMvcConfigurerAdapter {

@Override

**public** **void** addInterceptors(InterceptorRegistry registry) {

HandlerInterceptor handlerInterceptor=**new** HandlerInterceptor() {

@Override

**public** **boolean** preHandle(HttpServletRequest arg0, HttpServletResponse arg1, Object arg2) **throws** Exception {

System.***out***.println("自定义拦截器.....");

**return** **true**;

}

@Override

**public** **void** postHandle(HttpServletRequest arg0, HttpServletResponse arg1, Object arg2, ModelAndView arg3)

**throws** Exception {

// **TODO** Auto-generated method stub

}

@Override

**public** **void** afterCompletion(HttpServletRequest arg0, HttpServletResponse arg1, Object arg2, Exception arg3)

**throws** Exception {

// **TODO** Auto-generated method stub

}

};

registry.addInterceptor(handlerInterceptor).addPathPatterns("/\*\*");

}

**20.定义全局异常处理器：**

创建一个全局异常处理类，如下：

@ControllerAdvice

**public** **class** GlobalExceptionHandler {

@ExceptionHandler(Exception.**class**)

@ResponseBody

**public** Map<String,Object> allExceptionHandler(

Exception exception) **throws** Exception

{

Map<String,Object> map = **new** HashMap<String,Object>();

map.put("errorCode", 500);

map.put("errorMsg", exception.toString());

**return** map;

}

}

**21.异步调用:**

在项目中，当访问其它接口较慢或者做耗时任务时，不想程序一直卡在耗时任务上，想程序能够并行执行，我们可以使用多线程来并行的处理任务，SpringBoot提供了异步处理方式@Async。

@Service

**public** **class** AsyncServiceImpl **implements** AsyncService {

**public** **static** Random *random* =**new** Random();

@Async

@Override

**public** Future<String> doTask1() **throws** Exception {

System.***out***.println("开始做任务一");

**long** start = System.*currentTimeMillis*();

Thread.*sleep*(*random*.nextInt(10000));

**long** end = System.*currentTimeMillis*();

System.***out***.println("完成任务一，耗时：" + (end - start) + "毫秒");

**return** **new** AsyncResult<>("任务一完成");

}

@Async

@Override

**public** Future<String> doTask2()**throws** Exception {

System.***out***.println("开始做任务二");

**long** start = System.*currentTimeMillis*();

Thread.*sleep*(*random*.nextInt(10000));

**long** end = System.*currentTimeMillis*();

System.***out***.println("完成任务二，耗时：" + (end - start) + "毫秒");

**return** **new** AsyncResult<>("任务二完成");

}

@Async

@Override

**public** Future<String> doTask3() **throws** Exception{

System.***out***.println("开始做任务三");

**long** start = System.*currentTimeMillis*();

Thread.*sleep*(*random*.nextInt(10000));

**long** end = System.*currentTimeMillis*();

System.***out***.println("完成任务三，耗时：" + (end - start) + "毫秒");

**return** **new** AsyncResult<>("任务三完成");

}

}

在Controller中调用service功能:

@Controller

**public** **class** TestController {

@Autowired

**private** AsyncService asyncService;

@RequestMapping("/async")

@ResponseBody

**public** String getEntityById() **throws** Exception {

**long** start = System.*currentTimeMillis*();

Future<String> task1 = asyncService.doTask1();

Future<String> task2 = asyncService.doTask2();

Future<String> task3 = asyncService.doTask3();

**while**(**true**) {

**if**(task1.isDone() && task2.isDone() && task3.isDone()) {

// 三个任务都调用完成，退出循环等待

**break**;

}

Thread.*sleep*(1000);

}

**long** end = System.*currentTimeMillis*();

**return** "任务全部完成，总耗时：" + (end - start) + "毫秒";

}

}

在启动类上加入@EnableAsync注解。

**22.SpringBoot整合JSP：**

Spring Boot官方不推荐使用jsp，因为jsp相对于一些模板引擎，性能都比较低，官方推荐使用thymeleaf。

<!-- springboot整合jsp，需要是war工程，另外需要依赖另外两个包 -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-tomcat</artifactId>

</dependency>

<dependency>

<groupId>org.apache.tomcat.embed</groupId>

<artifactId>tomcat-embed-jasper</artifactId>

</dependency>

<!-- maven项目packaging为war类型时，必须要加这个插件 -->

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-war-plugin</artifactId>

<version>2.3</version>

<configuration>

<failOnMissingWebXml>false</failOnMissingWebXml>

</configuration>

</plugin>

</plugins>

</build>

在全局配置文件中：

spring.mvc.view.prefix=/WEB-INF/jsp/

spring.mvc.view.suffix=.jsp

**23.SpringBoot整合Freemarker:**

<!-- springboot不建议使用jsp，使用模板引擎，

比如themleaf,velocity,freemarker

整合freemarker -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-freemarker</artifactId>

</dependency>

#springboot整合freemarker

spring.freemarker.allow-request-override=false

spring.freemarker.cache=true

spring.freemarker.check-template-location=true

spring.freemarker.charset=UTF-8

spring.freemarker.content-type=text/html

spring.freemarker.expose-request-attributes=false

spring.freemarker.expose-session-attributes=false

spring.freemarker.expose-spring-macro-helpers=false

spring.freemarker.suffix=.ftl

spring.freemarker.template-loader-path=classpath:/templates

1. **SpringBoot整合Thymeleaf:**

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-thymeleaf</artifactId>

</dependency>

#springboot整合thymeleaf

#<!-- 关闭thymeleaf缓存 开发时使用 否则没有实时画面-->

spring.thymeleaf.cache=false

## 检查模板是否存在，然后再呈现

spring.thymeleaf.check-template-location=true

#Content-Type值

spring.thymeleaf.content-type=text/html

#启用MVC Thymeleaf视图分辨率

spring.thymeleaf.enabled=true

## 应该从解决方案中排除的视图名称的逗号分隔列表

spring.thymeleaf.excluded-view-names=

#模板编码

spring.thymeleaf.mode=LEGACYHTML5

# 在构建URL时预先查看名称的前缀

spring.thymeleaf.prefix=classpath:/templates/

# 构建URL时附加查看名称的后缀.

spring.thymeleaf.suffix=.html

# 链中模板解析器的顺序

#spring.thymeleaf.template-resolver-order= o

# 可以解析的视图名称的逗号分隔列表

#spring.thymeleaf.view-names=

#thymeleaf end

即声明thymeleaf使用非严 格的html。启动之后访问页面会报如下错误：

org.thymeleaf.exceptions.ConfigurationException: Cannot perform conversion to XML from legacy HTML: The nekoHTML library is not in classpath. nekoHTML 1.9.15 or newer is required for processing templates in "LEGACYHTML5" mode [http://nekohtml.sourceforge.net]. Maven spec: "net.sourceforge.nekohtml::nekohtml::1.9.15". IMPORTANT: DO NOT use versions of nekoHTML older than 1.9.15.

* 1

上面的异常已经说的很清楚了，需要依赖nekoHTML 1.9.15 or newer的版本。maven依赖如下

<dependency>

<groupId>net.sourceforge.nekohtml</groupId>

<artifactId>nekohtml</artifactId>

<version>1.9.22</version>

</dependency>

**25.SpringBoot整合JdbcTemplate:**

<!-- springboot整合jdbctemplate -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-jdbc</artifactId>

</dependency>

<!-- mysql的依赖 -->

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

</dependency>

连接数据库的配置：

spring.datasource.driver-class-name=com.mysql.jdbc.Driver

spring.datasource.username=root

spring.datasource.password=root

spring.datasource.url=jdbc:mysql://localhost:3306/db1

**25.SpringBoot整合QuartZ:**

//任务

@Component

**public** **class** MyJob {

@Scheduled(fixedRate=1000)

**public** **void** run()

{

System.***out***.println(**new** SimpleDateFormat("yyyy-MM-dd HH:mm:ss").format(**new** Date()));

}

}

在启动类上使用注解:@EnableScheduling

**SpringBoot整合MyBatis(xml配置方式):**

<!-- springboot整合mybatis -->

<dependency>

<groupId>org.mybatis.spring.boot</groupId>

<artifactId>mybatis-spring-boot-starter</artifactId>

<version>1.3.1</version>

</dependency>

<!-- MySQL -->

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

</dependency>

<!-- alibaba的druid数据库连接池 -->

<dependency>

<groupId>com.alibaba</groupId>

<artifactId>druid-spring-boot-starter</artifactId>

<version>1.1.0</version>

</dependency>

<dependency>

<groupId>com.alibaba</groupId>

<artifactId>druid</artifactId>

<version>1.0.11</version>

</dependency>

<!-- 分页插件 -->

<dependency>

<groupId>com.github.pagehelper</groupId>

<artifactId>pagehelper-spring-boot-starter</artifactId>

<version>1.1.2</version>

</dependency>

<!-- @Param注解在该包中 -->

<dependency>

<groupId>org.apache.ibatis</groupId>

<artifactId>ibatis-core</artifactId>

<version>3.0</version>

</dependency>

数据源配置：

spring:

datasource:

name: test

url: jdbc:mysql://127.0.0.1:3306/db1

username: root

password: root

type: com.alibaba.druid.pool.DruidDataSource

driver-class-name: com.mysql.jdbc.Driver

filters: stat

maxActive: 20

initialSize: 1

maxWait: 60000

minIdle: 1

timeBetweenEvictionRunsMillis: 60000

minEvictableIdleTimeMillis: 300000

validationQuery: select 'x'

testWhileIdle: **true**

testOnBorrow: **false**

testOnReturn: **false**

poolPreparedStatements: **true**

maxOpenPreparedStatements: 20

Mybatis配置文件:

<?xml version=*"1.0"* encoding=*"UTF-8"* ?>

<!DOCTYPE configuration

PUBLIC "-//mybatis.org//DTD Config 3.0//EN"

"http://mybatis.org/dtd/mybatis-3-config.dtd">

<configuration>

</configuration>

Mybatis的配置：

mybatis:

mapper-locations: classpath:mapping/UsersMapper.xml

#type-aliases-package: com.db1.pojo

config-location: classpath:mybatis/mybatis-config.xml

PageHelper分页插件:

pagehelper:

helperDialect: mysql

reasonable: **true**

supportMethodsArguments: **true**

params: count=countSql

**SpringBoot整合Mybatis(注解方式):**

<!-- springboot整合mybatis,注解版 -->

<dependency>

<groupId>org.mybatis.spring.boot</groupId>

<artifactId>mybatis-spring-boot-starter</artifactId>

<version>1.3.1</version>

</dependency>

<!-- MySQL -->

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

</dependency>

spring.datasource.driver-class-name=com.mysql.jdbc.Driver

spring.datasource.username=root

spring.datasource.password=root

spring.datasource.url=jdbc:mysql://localhost:3306/db1

**使用FastJson解析Json数据:**

SpringBoot默认配置的是Jackson。

使用FastJson解析JsonS数据:

<!-- fastjson的依赖 -->

<dependency>

<groupId>com.alibaba</groupId>

<artifactId>fastjson</artifactId>

<version>1.2.15</version>

</dependency>

**配置FastJson有两种方式：**

第一种：让启动类继承WebMvcConfigurerAdapter

void configureMessageConverters(List<HttpMessageConverter<?>> converters) {

FastJsonHttpMessageConverter convert = **new** FastJsonHttpMessageConverter();

FastJsonConfig config = **new** FastJsonConfig();

config.setSerializerFeatures(SerializerFeature.***PrettyFormat***);

convert.setFastJsonConfig(config);

converters.add(convert);

}

乱码解决：把springboot的response编码设置为utf-8这个功能开启就好

spring.http.encoding.force=true

第二种：@Bean注入

@Bean

**public** HttpMessageConverters fastJsonHttpMessageConverters()

{

FastJsonHttpMessageConverter convert = **new** FastJsonHttpMessageConverter();

FastJsonConfig config = **new** FastJsonConfig();

config.setSerializerFeatures(SerializerFeature.***PrettyFormat***);

convert.setFastJsonConfig(config);

HttpMessageConverter<?> con=convert;

**return** **new** HttpMessageConverters(con);

}

**SpringBoot区分多数据源分析：**

在项目中,我们通常会进行数据库拆分或是引入其他数据库，从而我们需要配置多个数据源:

如何区分多个数据源：

1. 通过包来区分:com.db1.mapper com.db2.mapper
2. 使用注解来区分:

**SpringBoot区分多数据源实现：**

<!-- springboot集成mybatis -->

<dependency>

<groupId>org.mybatis.spring.boot</groupId>

<artifactId>mybatis-spring-boot-starter</artifactId>

<version>1.3.1</version>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

</dependency>

#SpringBoot整合多数据源

spring.datasource.db1.driver-class-name=com.mysql.jdbc.Driver

spring.datasource.db1.username=root

spring.datasource.db1.password=root

spring.datasource.db1.url=jdbc:mysql://localhost:3306/db1

spring.datasource.db5.driver-class-name=com.mysql.jdbc.Driver

spring.datasource.db5.username=root

spring.datasource.db5.password=root

spring.datasource.db5.url=jdbc:mysql://localhost:3306/db5

为每个数据源进行配置:

@Configuration//注册到Spring容器中

@MapperScan(basePackages="com.borun.db1.mapper",sqlSessionFactoryRef="db1SqlSessionFactory")

**public** **class** DataSource1 {

//private DataSource1(){}

/\*\*

\* 配置db1数据库

\* **@return**

\*/

@Bean(name="db1Datasource")

@ConfigurationProperties(prefix="spring.datasource.db1")

**public** DataSource testDatasource() {

**return** DataSourceBuilder.*create*().build();

}

/\*\*

\* 创建SqlSessionFactory

\* **@param** dataSource

\* **@return**

\* **@throws** Exception

\*/

@Bean(name="db1SqlSessionFactory")

@Primary

**public** SqlSessionFactory testSqlSessionFactory(@Qualifier("db1Datasource")DataSource dataSource)

**throws** Exception {

SqlSessionFactoryBean bean=**new** SqlSessionFactoryBean();

bean.setDataSource(dataSource);

**return** bean.getObject();

}

/\*\*

\* 配置事务管理

\* **@param** dataSource

\* **@return**

\*/

@Bean(name="db1TransactionManager")

**public** DataSourceTransactionManager testTransactionManager(

@Qualifier("db1Datasource")DataSource dataSource) {

**return** **new** DataSourceTransactionManager(dataSource);

}

@Bean(name="db1SqlSessionTemplate")

**public** SqlSessionTemplate testSqlSessionTemplate(@Qualifier("db1SqlSessionFactory")

SqlSessionFactory sqlSessionFactory) {

**return** **new** SqlSessionTemplate(sqlSessionFactory);

}

}

@Configuration

@MapperScan(basePackages="com.borun.db5.mapper",sqlSessionFactoryRef="db5SqlSessionFactory")

**public** **class** DataSource2 {

/\*\*

\* 配置db5数据库

\* **@return**

\*/

@Bean(name="db5Datasource")

@ConfigurationProperties(prefix="spring.datasource.db5")

**public** DataSource testDatasource() {

**return** DataSourceBuilder.*create*().build();

}

/\*\*

\* 创建SqlSessionFactory

\* **@param** dataSource

\* **@return**

\* **@throws** Exception

\*/

@Bean(name="db5SqlSessionFactory")

**public** SqlSessionFactory testSqlSessionFactory(@Qualifier("db5Datasource")DataSource dataSource)

**throws** Exception {

SqlSessionFactoryBean bean=**new** SqlSessionFactoryBean();

bean.setDataSource(dataSource);

//如果还有分页等其他事务

// bean.setMapperLocations(new PathMatchingResourcePatternResolver().

// getResources("classpath:mybatis/test1/\*.xml"));

**return** bean.getObject();

}

/\*\*

\* 配置事务管理

\* **@param** dataSource

\* **@return**

\*/

@Bean(name="db5TransactionManager")

**public** DataSourceTransactionManager testTransactionManager(

@Qualifier("db5Datasource")DataSource dataSource) {

**return** **new** DataSourceTransactionManager(dataSource);

}

@Bean(name="db5SqlSessionTemplate")

**public** SqlSessionTemplate testSqlSessionTemplate(@Qualifier("db5SqlSessionFactory")

SqlSessionFactory sqlSessionFactory) {

**return** **new** SqlSessionTemplate(sqlSessionFactory);

}

}

**事务管理:**

在Spring Boot中推荐使用@Transactional注解来声明事务。

当我们使用了spring-boot-starter-jdbc或spring-boot-starter-data-jpa依赖的时候，SpringBoot会自动默认分别注入DataSourceTransactionManager或JpaTransactionManager

**SpringBoot整合JPA-Hibernate:**

<!-- springboot整合jpa -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

</dependency>

全局配置配置文件：

spring.datasource.driver-class-name=com.mysql.jdbc.Driver

spring.datasource.username=root

spring.datasource.password=root

spring.datasource.url=jdbc:mysql://localhost:3306/db6

#让控制器输出json字符串格式

spring.jackson.serialization.indent-output=true

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.jpa.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect

**SpringBoot整合Email:**

<!-- 邮件依赖 -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-mail</artifactId>

</dependency>

全局配置文件:

# JavaMailSender 邮件发送的配置

spring.mail.host=smtp.qq.com #smtp.163.com

spring.mail.username=465008297@qq.com

spring.mail.password=cjmegwewmbiqcach #授权码

spring.mail.properties.mail.smtp.auth=true

spring.mail.properties.mail.smtp.starttls.enable=true

spring.mail.properties.mail.smtp.starttls.required=true

#若使用QQ邮箱发送邮件，则需要修改为spring.mail.host=smtp.qq.com，同时spring.mail.password改为QQ邮箱的授权码。

#QQ邮箱->设置->账户->POP3/SMTP服务:开启服务后会获得QQ的授权码

出现认证失败的解决方案:因为JDK1.8中jre\lib\security中两个 jar 包替换的缘故。将下载后的local\_policy.jar和US\_export\_policy.jar替换到JDK1.8的jre\lib\security文件夹即可。

发送带附件的邮件：

**public** **void** sendAttachmentMail(String sendTo, String title, String content, File file) {

MimeMessage mimeMessage = mailSender.createMimeMessage();

**try** {

MimeMessageHelper helper = **new** MimeMessageHelper(mimeMessage, **true**);

helper.setFrom(emailConfig.getEmailFrom());

helper.setTo(sendTo);

helper.setSubject(title);

helper.setText(content);

FileSystemResource r = **new** FileSystemResource(file);

helper.addAttachment("附件", r);

} **catch** (Exception e) {

e.printStackTrace();

}

mailSender.send(mimeMessage);

}

发送模板邮件:

@Override

**public** **void** sendTemplateMail(String sendTo, String title,String file) {

MimeMessage message = mailSender.createMimeMessage();

**try** {

MimeMessageHelper helper = **new** MimeMessageHelper(message, **true**);

helper.setFrom(emailConfig.getEmailFrom());

helper.setTo(sendTo);

helper.setSubject(title);

Map<String, Object> model = **new** HashMap();

model.put("username", "小华");

//修改 application.properties 文件中的读取路径

// FreeMarkerConfigurer configurer = new FreeMarkerConfigurer();

// configurer.setTemplateLoaderPath("classpath:templates");

//读取 html 模板

Template template = freeMarkerConfigurer.getConfiguration().getTemplate(file);

String html = FreeMarkerTemplateUtils.*processTemplateIntoString*(template, model);

helper.setText(html, **true**);

} **catch** (Exception e) {

e.printStackTrace();

}

mailSender.send(message);

}

**SpringBoot实现文件上传:**

<!-- spring boot web支持 -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<!-- thymeleaf模板依赖-->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-thymeleaf</artifactId>

</dependency>

multipart.maxFileSize=500Mb

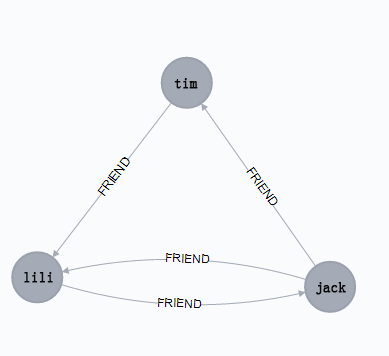
multipart.maxRequestSize=500Mb

**SpringBoot实现批量文件上传:**

**Neo4j:**

Neo4j是一个高性能的，NOSQL图形数据库，它是一个嵌入式的、基于磁盘的、具备完全的事务特性的Java持久化引擎，但是它将结构化数据存储在网络(从数学角度叫做图)上而不是表中。Neo4j也可以被看作是一个高性能的图引擎，该引擎具有成熟数据库的所有特性。

这种数据库与传统的关系型数据库有很大的差别。为了更好地帮助大家理解我这里就将关系型数据库与图形数据库作个比较。   
关系型数据库：   
我们常用的像mysql，oracle等都是关系型数据库，在关系型数据库里面对数据的处理是这样子的：对每个对象都创建一个表，对象的属性对应表里面的列   
IMG_256  
如图所示。在数据库里有条数据表示两个对象：tim,lili。在现实生活中我们会发现任何对象都是有某种联系的，那么关系型数据库里是怎样来表示这种关系呢?就比如tim和lili是好朋友，那么在数据库里怎样来表示他们的关系呢?关系型数据库里面是 这样处理的——新建一个叫relationship的表，表里面有两个字段id，friendid。IMG_257   
  
如图所示。如果我们要查找tim的朋友那么我们可以遍历relationship表就可以了。   
这种数据模型会有什么问题呢？其实我们可以对这个数据模型提个问题——tim的朋友的朋友的朋友的朋友是谁？好，关系型数据会这样回答你的问题：首先在relationship表里面找到所有id为1(tim的id)的数据，然后拿到对应的friendid，接着逐个根据friendid再进行遍历找到对应的friendid，如此反复地遍历查询。。。。   
也许10分钟也许一小时，也许它永远都无法回答你的问题。   
其实，这种关系只要超过5级关系型数据库就无法解决问题，这就是为什么需要图形数据库的出现了。   
图形数据库：   
在图形数据库里面对数据的处理是这样子的：每个对象都表示成为一个节点（node），每个节点之间的联系表示成关系（relationship），节点与节点之间用关系关联在一起。我们可以看图更好理解一点。

  
如图所示，有三个节点（node）它们都通过FRIEND关系（relationship）关联起来。Tim的朋友是lili,lili跟jack互为朋友，同时jack认识tim。在图形数据库里要回答像“tim的朋友的朋友的朋友”的问题非常简单，数据库只需要找到tim的关系（relationship）所对应的节点然后找到对应节点的关系（relationship），只需遍历几次，这样就可以很容易回答了上面的问题了。

1. 下载windows环境下的安装包neo4j-community-3.3.4-windows.zip

2. 解压，比如目录为E:\neo4j

3. 设置环境变量

变量名：NEO4J\_HOME

变量值：E:\neo4j

再修改变量path，增加%NEO4J\_HOME%\bin

4.查看并编辑配置参数

%NEO4J\_HOME%\conf\neo4j.conf

5.启动

在DOS命令行窗口，执行：neo4j.bat console

6. 打开neo4j集成的浏览器

<http://localhost:7474/>

# Cypher语言:Cypher是专门为图像数据库设计的语言，它浅显易懂。举个例子，假如要创建这么一个数据模型——好友A与好友B是朋友关系，好友B与C也是朋友关系，好友A认识C但C不认识A——这样的数据模型用Cypher描述如下:

create (A:Person｛name:'jack'})-[:Friend]->(B:Person {name:'tom'})-[:Friend]->(C:Person {name:'lucy'}), (A)-[:Know]->(C)

(注:（）里面的是节点，［］里面的是关系，{}里的是属性, >表示关系的方向)

要查询A的一个名叫”tom”的朋友:

match （a)-[:Friend]->(b)

where b.name='tom'

return b

**SpringBoot整合Neo4j:**

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-neo4j</artifactId>

</dependency>

配置文件中：

spring.data.neo4j.uri=http://localhost:7474

spring.data.neo4j.username=neo4j

spring.data.neo4j.password=123

@Component

**public** **interface** UserRepository **extends** GraphRepository<UserNode> {

@Query("MATCH (n:User) RETURN n ")

List<UserNode> getUserNodeList();

@Query("create (n:User{age:{age},name:{name}}) RETURN n ")

List<UserNode> addUserNodeList(@Param("name") String name, @Param("age")**int** age);

}

@Component

**public** **interface** UserRelationRepository **extends** GraphRepository<UserRelation> {

@Query("match p=(n:User)<-[r:UserRelation]->(n1:User) where n.userId={firstUserId} and n1.userId={secondUserId} return p")

List<UserRelation> findUserRelationByEachId(@Param("firstUserId") String firstUserId, @Param("secondUserId") String secondUserId);

@Query("match (fu:User),(su:User) where fu.userId={firstUserId} and su.userId={secondUserId} create p=(fu)-[r:UserRelation]->(su) return p")

List<UserRelation> addUserRelation(@Param("firstUserId") String firstUserId, @Param("secondUserId") String secondUserId);

}

**SpringBoot整合Redis(单机版):**

<!-- springboot整合redis -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-redis</artifactId>

</dependency>

#redis单服务器配置

spring.redis.database=0

spring.redis.host=192.168.25.128

spring.redis.port=6379

spring.redis.pool.max-active=8

spring.redis.pool.max-wait=-1

spring.redis.pool.max-idle=8

spring.redis.pool.min-idle=0

spring.redis.timeout=0

**SpringBoot整合Redis(集群版):**

<!-- springboot整合redis -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-redis</artifactId>

</dependency>

spring.redis.cluster.nodes=192.168.22.12:6666

**SpringBoot整合ActiveMQ:**

<!-- 整合ActiveMQ的依赖 -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-activemq</artifactId>

</dependency>

spring.activemq.broker-url=tcp://192.168.25.129:61616

spring.activemq.in-memory=true

spring.activemq.pool.enabled=false

spring.activemq.user=admin

spring.activemq.password=admin

#如果此处设置为true，需要加如下面的依赖包，否则会自动配置失败，JmsMessagingTemplate

spring.activemq.pool.enabled=false

<dependency>

<groupId>org.apache.activemq</groupId>

<artifactId>activemq-pool</artifactId>

</dependency>

**SpringBoot整合RabbitMQ:**

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-amqp</artifactId>

</dependency>

全局配置:

spring.application.name=sbrabbitmq

spring.rabbitmq.addresses=192.168.25.129:5672

spring.rabbitmq.username=admin

spring.rabbitmq.password=admin

spring.rabbitmq.publisher-confirms=true

发消息:

@Component

**public** **class** HelloSender {

@Autowired

**private** AmqpTemplate rabbitTemplate;

**public** String send() {

String context = "hello " + ",你好";

**this**.rabbitTemplate.convertAndSend("hel", context);

**return** "发送成功";

}

}

收消息:

@Component

**public** **class** HelloReceiver {

@RabbitHandler

@RabbitListener(queues = "hel")

**public** **void** process(String text) {

System.***out***.println("接收到消息 : " + text);

}

}

**Kafka概述及安装:**

Apache Kafka是一个分布式发布 - 订阅消息系统和一个强大的队列，可以处理大量的数据，并使您能够将消息从一个端点传递到另一个端点。 Kafka适合离线和在线消息消费。 Kafka消息保留在磁盘上，并在群集内复制以防止数据丢失。 Kafka构建在ZooKeeper同步服务之上。 它与Apache Storm和Spark非常好地集成，用于实时流式数据分析。

以下是Kafka的几个好处 -

**可靠性** - Kafka是分布式，分区，复制和容错的。

**可扩展性** - Kafka消息传递系统轻松缩放，无需停机。

**耐用性** - Kafka使用分布式提交日志，这意味着消息会尽可能快地保留在磁盘上，因此它是持久的。

**性能** - Kafka对于发布和订阅消息都具有高吞吐量。 即使存储了许多TB的消息，它也保持稳定的性能。Kafka非常快，并保证零停机和零数据丢失。

**SpringBoot整合Kafka:**

<dependency>

<groupId>org.springframework.kafka</groupId>

<artifactId>spring-kafka</artifactId>

<version>1.1.1.RELEASE</version>

</dependency>

全局配置:

spring:

kafka:

producer:

bootstrap-servers: 192.168.25.130:9092

consumer:

group-id: springboot-group1

auto-offset-reset: earliest

发消息:

@Component

@EnableScheduling

**public** **class** KafkaProducer {

@Autowired

**private** KafkaTemplate kafkaTemplate;

/\*\*

\* 定时任务

\*/

@Scheduled(cron = "00/1 \* \* \* \* ?")

**public** **void** send(){

String message = UUID.*randomUUID*().toString();

ListenableFuture future = kafkaTemplate.send("tests", message);

future.addCallback(o -> System.***out***.println("send-消息发送成功：" + message), throwable -> System.***out***.println("消息发送失败：" + message));

}

}

**SpringBoot整合Solr：**

<dependency>

<groupId>org.springframework.data</groupId>

<artifactId>spring-data-solr</artifactId>

</dependency>

全局配置:

spring.data.solr.host=http://192.168.25.129:8080/solr

访问solr：

@Controller

**public** **class** SolrController {

@Autowired

**private** SolrClient solrClient;

@RequestMapping("/solr")

@ResponseBody

**public** String testSolr() **throws** SolrServerException, IOException

{

SolrDocument doc = solrClient.getById("562379");

**return** doc.toString();

}

}

**Elasticsearch概述及安装:**

ElasticSearch是一个基于Lucene的搜索服务器。它提供了一个分布式多用户能力的全文搜索引擎，基于RESTful web接口。Elasticsearch是用Java开发的，并作为Apache许可条款下的开放源码发布，是当前流行的企业级搜索引擎。用于[云计算](https://baike.so.com/doc/580575-614558.html)中，能够达到实时搜索，稳定，可靠，快速，安装使用方便。

**安装:**

1.首先安装好jdk.

2. 从Elastic的官网下载ES的安装包

<https://download.elastic.co/elasticsearch/release/org/elasticsearch/distribution/tar/elasticsearch/2.4.4/elasticsearch-2.4.4.tar.gz>

3.解压到/usr/local下

tar -xzvf elasticsearch-2.4.4.tar.gz -C /usr/local

4.编辑配置文件:/config/elasticsearch.yml

bootstrap.memory\_lock: true

index.cache.field.max\_size: 50000

index.cache.field.expire: 30m

index.cache.field.type: soft

network.host: 192.168.\*.25,10.29.\*.58,127.0.0.1

action.disable\_delete\_all\_indices : true

5.启动：启动时不要使用root用户，会报错java.lang.RuntimeException: don't run elasticsearch as root

使用普通用户启动会显示权限不够，解决办法：

将ElasticSearch的安装目录及其子目录改为另外一个非root账户，如：

sudo chown -R linux elasticsearch-2.4.4

sudo chgrp -R linux elasticsearch-2.4.4

**在Elasticsearch中存储数据**：在Elasticsearch中，所有的数据，都以文档的形式存储，每个文档都有定义好的索引和类型。每个文档可以包含一个或多个字段来保存数据。

在sence中，输入如下的curl请求代码:

curl -XPUT '192.168.25.129:9200/userindex/user/3?pretty' -d'

{

"first\_name" : "Douglas",

"last\_name" : "Fir",

"age" : 23,

"about": "I like to build cabinets",

"interests": [ "forestry" ]

}'

**从Elasticsearch中检索文档:**

在sence中，输入如下的curl请求代码

curl -XGET "http://192.168.25.129:9200/userindex/user/1"

搜索所有用户:

curl -XGET "http://192.168.25.129:9200/userindex/user/\_search"

curl -XGET "http://192.168.25.129:9200/userindex/user/\_search?q=age:32"

**SpringBoot整合ElasticSearch:**

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-elasticsearch</artifactId>

</dependency>

application.yml:

spring:

data:

elasticsearch:

cluster-nodes: 192.168.25.129:9300

local: **false**

repositories:

enabled: **true**

封装实体类:

@Document(indexName="userindex",type="user")

**public** **class** User **implements** Serializable{

@Id

**private** Long id;

**private** String first\_name;

**private** String last\_name;

**private** **int** age;

**private** String about;

**private** List<String> interests;}

**SpringBoot整合Shiro:**

插入测试数据**:**

INSERT INTO `sys\_permission` (`id`,`available`,`name`,`parent\_id`,`parent\_ids`,`permission`,`resource\_type`,`url`) VALUES (1,0,'用户管理',0,'0/','userInfo:view','menu','userInfo/userList');

INSERT INTO `sys\_permission` (`id`,`available`,`name`,`parent\_id`,`parent\_ids`,`permission`,`resource\_type`,`url`) VALUES (2,0,'用户添加',1,'0/1','userInfo:add','button','userInfo/userAdd');

INSERT INTO `sys\_permission` (`id`,`available`,`name`,`parent\_id`,`parent\_ids`,`permission`,`resource\_type`,`url`) VALUES (3,0,'用户删除',1,'0/1','userInfo:del','button','userInfo/userDel');

INSERT INTO `sys\_role` (`id`,`available`,`description`,`role`) VALUES (1,'0','管理员','admin');

INSERT INTO `sys\_role` (`id`,`available`,`description`,`role`) VALUES (2,'0','VIP会员','vip');INSERT INTO `sys\_role\_permission` VALUES ('1', '1');

INSERT INTO `sys\_role\_permission` (`permission\_id`,`role\_id`) VALUES (1,1);

INSERT INTO `sys\_role\_permission` (`permission\_id`,`role\_id`) VALUES (1,2);

INSERT INTO `sys\_role\_permission` (`permission\_id`,`role\_id`) VALUES (1,3);

INSERT INTO `sys\_user\_role` (`role\_id`,`uid`) VALUES (1,1);

INSERT INTO `sys\_user` (`uid`,`username`,`name`,`password`,`salt`,`state`) VALUES ('1', 'admin', '管理员', 'd3c59d25033dbf980d29554025c23a75', '8d78869f470951332959580424d4bf4f', 0);

### Shiro 配置:

**SpringBoot整合Mongodb:**

MongoDB是一个基于分布式文件存储的数据库。由C++语言编写。旨在为WEB应用提供可扩展的高性能数据存储解决方案。MongoDB是一个介于[关系数据库](https://baike.so.com/doc/2023251-2140958.html)和非关系数据库之间的产品，是非关系数据库当中功能最丰富，最像关系数据库的。他支持的数据结构非常松散，是类似[json](https://baike.so.com/doc/663437-702310.html)的bson格式，因此可以存储比较复杂的数据类型。Mongo最大的特点是他支持的查询语言非常强大，其语法有点类似于面向对象的查询语言，几乎可以实现类似关系数据库单表查询的绝大部分功能，而且还支持对数据建立索引。

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-mongodb</artifactId> </dependency>

配置:

spring:

data:

mongodb:

host: 192.168.25.130

port: 27017

database: mydb

**SpringBoot整合WebSocket:**

HTTP 协议是一种无状态的、无连接的、单向的应用层协议。它采用了请求/响应模型。通信请求只能由客户端发起，服务端对请求做出应答处理。这种通信模型有一个弊端：HTTP 协议无法实现服务器主动向客户端发起消息。

这种单向请求的缺点，如果服务器有连续的状态变化，客户端要获知就非常麻烦。大多数 Web 应用程序将通过频繁的异步JavaScript和XML（AJAX）请求实现长轮询。轮询的效率低，非常浪费资源（因为必须不停连接，或者 HTTP 连接始终打开）。

WebSocket 连接允许客户端和服务器之间进行全双工通信，以便任一方都可以通过建立的连接将数据推送到另一端。WebSocket 只需要建立一次连接，就可以一直保持连接状态。这相比于轮询方式的不停建立连接显然效率要大大提高。

依赖:

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-websocket</artifactId>

</dependency>

WebSocket的配置类:

//WebSocket的配置类

@Configuration

//开启对WebSocket的支持

@EnableWebSocketMessageBroker

**public** **class** WebSocketConfig **extends** AbstractWebSocketMessageBrokerConfigurer{

//注册一个STOMP协议的节点，并映射到指定的URL

@Override

**public** **void** registerStompEndpoints(StompEndpointRegistry registry) {

//注册一个STOMP的endpoint,并指定使用SockJS协议

registry.addEndpoint("/endpointSocket").withSockJS();

}

//配置消息代理

@Override

**public** **void** configureMessageBroker(MessageBrokerRegistry registry) {

//配置一个广播式的消息代理

registry.enableSimpleBroker("/topic");

}

}

//浏览器向服务器发送消息实体类

**public** **class** SocketMessage {

**private** String message;

**public** String getMessage() {

**return** message;

}

**public** **void** setMessage(String message) {

**this**.message = message;

}

}

//服务器向浏览器响应数据的封装实体类

**public** **class** SocketResponse {

**private** String responseMessage;

**public** SocketResponse(String responseMessage) {

**this**.responseMessage = responseMessage;

}

**public** String getResponseMessage() {

**return** responseMessage;

}

}

控制类：

@Controller

**public** **class** WebSocketController {

**private** SimpleDateFormat sf = **new** SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

//当浏览器向服务器端发送STOMP请求时，通过@MessageMapping注解来映射/getServerTime地址

@MessageMapping(value = "/getServerTime")

//当服务端有消息时，会对订阅了@SendTo中的路径的客户端发送消息

@SendTo(value = "/topic/getResponse")

**public** SocketResponse serverTime(SocketMessage message) **throws** InterruptedException {

**return** **new** SocketResponse(message.getMessage() + sf.format(**new** Date()));

}

}

**SpringBoot整合AngularJS:**

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-thymeleaf</artifactId>

</dependency>

<dependency>

<groupId>net.sourceforge.nekohtml</groupId>

<artifactId>nekohtml</artifactId>

<version>1.9.22</version>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-jdbc</artifactId>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

</dependency>

<dependency>

<groupId>org.apache.commons</groupId>

<artifactId>commons-lang3</artifactId>

<version>3.3.2</version>

</dependency>

全局配置:

spring.datasource.driver-class-name=com.mysql.jdbc.Driver

spring.datasource.username=root

spring.datasource.password=root

spring.datasource.url=jdbc:mysql://localhost:3306/db1

#springboot整合thymeleaf

#<!-- 关闭thymeleaf缓存 开发时使用 否则没有实时画面-->

spring.thymeleaf.cache=false

## 检查模板是否存在，然后再呈现

spring.thymeleaf.check-template-location=true

#Content-Type值

spring.thymeleaf.content-type=text/html

#启用MVC Thymeleaf视图分辨率

spring.thymeleaf.enabled=true

## 应该从解决方案中排除的视图名称的逗号分隔列表

spring.thymeleaf.excluded-view-names=

#模板编码

spring.thymeleaf.mode=LEGACYHTML5

# 在构建URL时预先查看名称的前缀

spring.thymeleaf.prefix=classpath:/templates/

# 构建URL时附加查看名称的后缀.

spring.thymeleaf.suffix=.html

# 链中模板解析器的顺序

#spring.thymeleaf.template-resolver-order= o

# 可以解析的视图名称的逗号分隔列表

#spring.thymeleaf.view-names=

#thymeleaf end

**打包发布到独立的tomcat:**

1:需要打成war包

2：<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-tomcat</artifactId>

<scope>provided</scope>

</dependency>

3：启动类继承SpringBootServletInitializer,重写configure方法