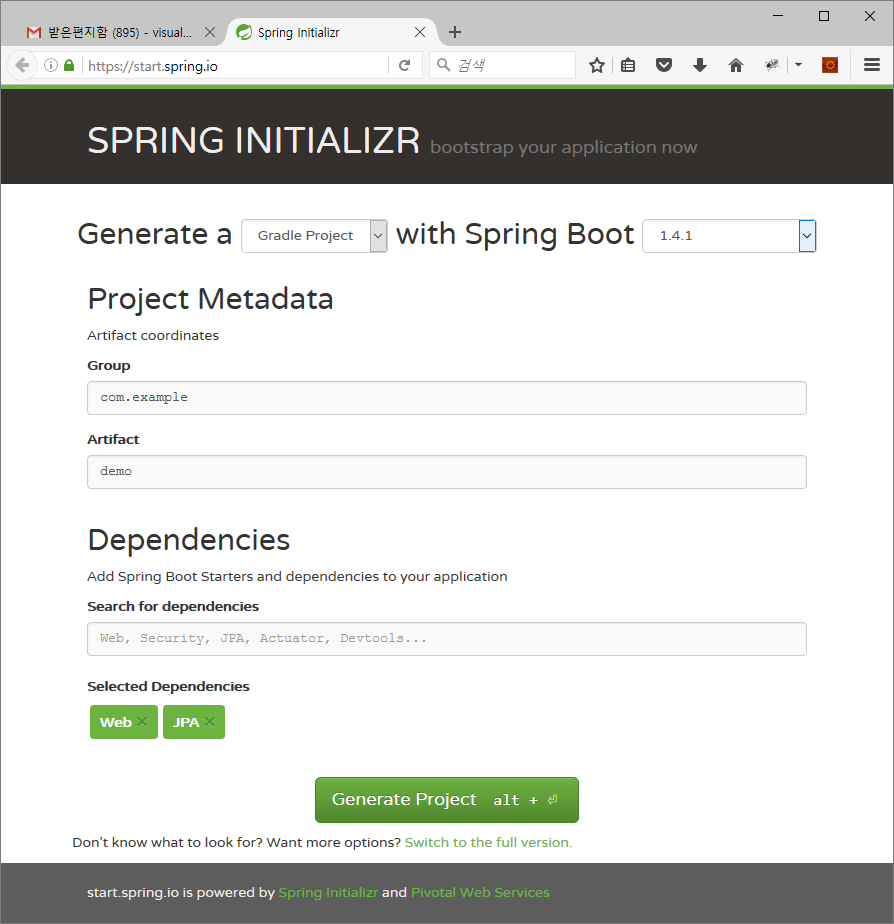
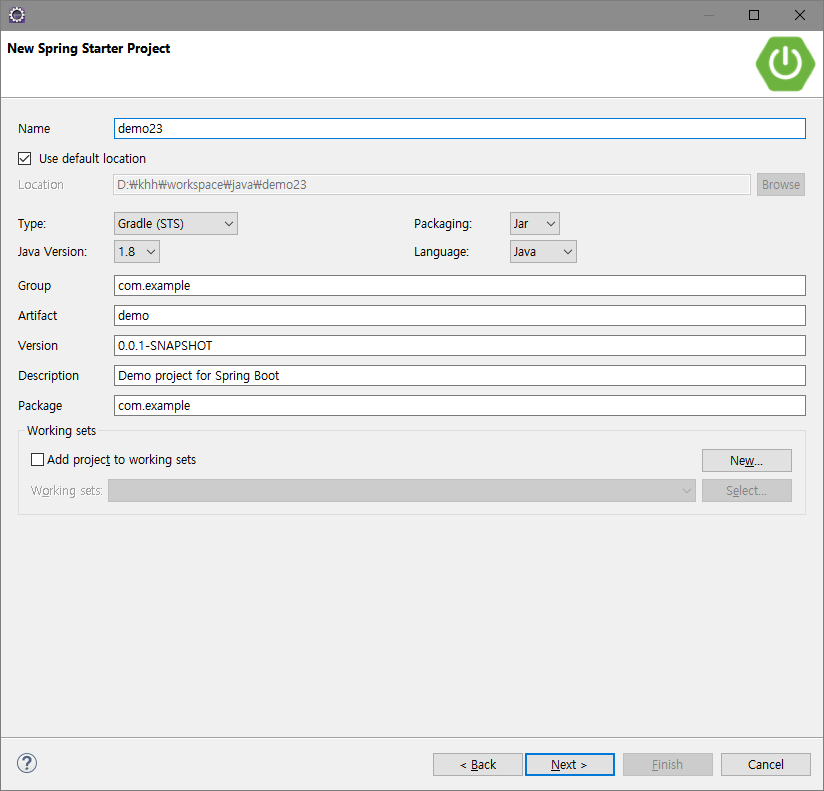
스프링 부트의 핵심

|  |  |
| --- | --- |
| 자동 구성 | 스프링부트는 많은 스프링 애플리케이션에 공통으로 필요한 애플리케이션 기능을 자동으로 구성한다 |
| 스타터 의존성 | 스프링 부트에 어떤 기능이 필요한지 알려 주면 필요한 라이브러리를 빌드에 추가한다는 것을 보장한다 |
| 명령줄 인터페이스 | 스프링 부트의 이 부가 기능을 이용하면 애플리케이션 코드만 작성해도 완전한 애플리케이션을 개발 할 수있지만, 기존 프로젝트 빌드 방식에는 필요 없는 기능이다. |
| 액추에이터 | 스프링 부트 애플리케이션을 실행할때 내부에서 어떤 일이 일어나는지 이해할수있다 |

부트시작



|  |  |
| --- | --- |
| **import** org.springframework.boot.SpringApplication;  **import** org.springframework.boot.autoconfigure.SpringBootApplication;  @SpringBootApplication  **public** **class** Demo2Application {  **public** **static** **void** main(String[] args) {  SpringApplication.*run*(Demo2Application.**class**, args);  }  } | |
| @SpringBootApplication | 세개를 묶은것이다.   |  |  | | --- | --- | | **@Configuration** | 스프링의 자바 기반 구성 클래스로 지정한다 책에서는 많은 구성 코드를 작성하지 않지만. 구성이 필요하다면 XML구성대신 자바 기반구성을 사용한다 | | **@ComponentScan** | 컴포넌트 검색 기능을 활성화해서 웹컨트롤러 클래스나 다른 컴포넌트 클래스들을 자동으로 검색하여 스프링 애플리케이션 컨텍스트에 빈으로 등록시킨다  MVC컨트롤러를 작성하려면 @Controller 어노테이션을 붙여 컴포넌트 검색이 이컨트롤러를 찾을수 있게할것이다. | | **@EnableAutoConfiguration** | 마법이라고 해도 무방하다 스프링 부트의 자동 구성 마법이 일어나기 때문이다 이것으로 수많은 구성 코드를 대처할수있다. | |

**부트 실행하기**

|  |
| --- |
| gradle bootRun |
| java -jar ...../asd.jar |

**스프링부트 애플리케이션 테스트**

|  |
| --- |
| **import** org.junit.Test;  **import** org.junit.runner.RunWith;  **import** org.springframework.boot.test.context.SpringBootTest;  **import** org.springframework.test.context.junit4.SpringRunner;  @RunWith(SpringRunner.**class**)  @SpringBootTest  **public** **class** Demo2ApplicationTests {  @Test  **public** **void** contextLoads() {  }  } |

**스프링 부트 프로젝트 빌드 파헤치기**

|  |
| --- |
| **buildscript** {  **ext** {  springBootVersion = '1.4.1.RELEASE'  }  **repositories** {  mavenCentral()  }  **dependencies** {  classpath("org.springframework.boot:spring-boot-gradle-plugin:${springBootVersion}")  }  }  **apply** plugin: 'java'  **apply** plugin: 'eclipse'  **apply** plugin: 'spring-boot'  jar {  baseName = 'demo'  **version** = '0.0.1-SNAPSHOT'  }  **sourceCompatibility** = 1.8  targetCompatibility = 1.8  **repositories** {  mavenCentral()  }  **dependencies** {  compile('org.springframework.boot:spring-boot-starter-data-jpa')  compile('org.springframework.boot:spring-boot-starter-web')  testCompile('org.springframework.boot:spring-boot-starter-test')  } |

**어플리케이션 기능에 집중하자**

**무슨일이 일어났을까?**

자신만의 조건을 작성하는 방법

Condition 인터페이스를 상속받고 matches()메서드를 오버라이드 하면된다. JdbcTemplate이 클래스패스에 있을때만 통과한다.

|  |
| --- |
| **import** org.springframework.context.annotation.Condition;  **import** org.springframework.context.annotation.ConditionContext;  **import** org.springframework.core.type.AnnotatedTypeMetadata;  **public** **class** CustomCondition **implements** **Condition** {  @Override  **public** **boolean** matches(ConditionContext context, AnnotatedTypeMetadata metadata) {  **try** {  context.getClassLoader().loadClass("org.springframework.jdbc.core.JdbcTemplate");  **return** **true**;  } **catch** (ClassNotFoundException e) {  **return** **false**;  }  }  } |
| @**Conditional**(JdbcTemplateContition.**class**)  @Bean  **public** **class** Myservice {  ...  } |
| JdbcTemplateCondition을 통과할떄만 Myservice빈을 생성한다. 그렇지않으면빈선언이 무효된다.  다시말해 JdbcTemplate가 클래스 패스에 있을때만 MyService빈을 생성한다. |

**자동구성에 사용되는 조건 어노테이션**

|  |  |
| --- | --- |
| @ConditionalOnBean | 대상빈을 구성함 |
| @ConditionalOnMissingBean | 대상 빈을 아직 구성하지 않음 |
| @ConditionalOnClass | 대상 클래스가 클래스 패스에 있음 |
| @ConditionalOnMissionClass | 대상클래스가 클래스 패스에 없음 |
| @ConditionalOnExpression | 스프링 표현식(언에(SpEL)가 참(true) |
| @ConditionalOnJava | 자바 버전이 특정 버전 또는 범위 버전 맞음 |
| @ConditionalOnJndi | JNDI InitialContext가 사용 가능하고 선택적으로 지정한 JNDI위치가 있음 |
| @ConditionalOnProperty | 지정한 구성 프로퍼티가 기대하는 값을 가짐 |
| @ConditionalOnResource | 지정한 리소스가 클래스패스에 있음 |
| @ConditionalOnWebApplication | 애플리케이션이 웹 애플리케이션임 |
| @ConditionalOnNotWebApplication | 애플리케이션이 웹 애플리케이션이 아님 |

예를들어 DataSourceAutoConfigration은 이런식으로 구성되어있다.

|  |
| --- |
| **package** org.springframework.boot.autoconfigure.jdbc;  @Configuration  @ConditionalOnClass({ DataSource.**class**, EmbeddedDatabaseType.**class** })  @EnableConfigurationProperties(DataSourceProperties.**class**)  @Import({ Registrar.**class**, DataSourcePoolMetadataProvidersConfiguration.**class** })  **public** **class** DataSourceAutoConfiguration {  **private** **static** **final** Log ***logger*** = LogFactory.*getLog*(DataSourceAutoConfiguration.**class**);  @Bean  @ConditionalOnMissingBean  **public** DataSourceInitializer dataSourceInitializer() {  **return** **new** DataSourceInitializer();  }.... |
| @ConditionalOnClass({ DataSource.**class**, EmbeddedDatabaseType.**class** })  **를등록 하였기때문에 클래스 패스에 DataSource, EmbeddeDatabaseType이 없으면 조건을 만족하지 못하니.DataSourceAutoConfigration이 제공하는 구성을 모두 무시한다.** |
|  |
| **package** org.springframework.boot.autoconfigure.jdbc;  @Configuration  @ConditionalOnClass({ DataSource.**class**, JdbcTemplate.**class** })  @ConditionalOnSingleCandidate(DataSource.**class**)  @AutoConfigureAfter(DataSourceAutoConfiguration.**class**)  **public** **class** JdbcTemplateAutoConfiguration {  **private** **final** DataSource dataSource;  **public** JdbcTemplateAutoConfiguration(DataSource dataSource) {  **this**.dataSource = dataSource;  }  @Bean  @Primary  @ConditionalOnMissingBean(JdbcOperations.**class**)  **public** JdbcTemplate jdbcTemplate() {  **return** **new** JdbcTemplate(**this**.dataSource);  } |
| @ConditionalOnMissingBean(JdbcOperations.**class**)  클래스 패스에 JdbcOperation가 없을시에 빈이 생성된다. |

1. H2가 클래스패스에 있으므로 내장 H2 데이터 베이스를 새엇ㅇ한다 생성하는 빈 타입은 javax.sql.DataSource 데이터 베이스에 접속 할때 필요하다

2. 하이버네이트 엔티티 매니저(EntityManager)가 스프링 데이터 JAP로 전이적으로) 클래스패이에 있으므로(스프링의 LocalContainerEntityManagerFactoryBean과 JapVendorAdapter를 포함하여) 하이버네이트를 다룰 때 필요한 빈을 자동 구성한다.

3. 스프링 데이터 JPA가 클래스 패으에 있으므로 리포지토리 인터페이스에서 구현체를 자동으로 생성하려고 스프링 데이터 JPA를 구성한다

4. Thymeleaf가 클래스패스에 있으므로 Thymeleaf가 스프링 MVC의 뷰 옵션으로 구성된다. (Thymeleaf 템플릿 리졸버, 템플릿 엔진, 뷰 리졸버 포함) 템플릿 리졸버는 클래스패스의 루트를 기준으로 /templates위치에 있는 템플릿을 해석하도록 구성한다.

5.웹 스타터 의존성 덕분에 스프링 MVC가 클래스 패스에 있으므로 스프링 DispatcherServlet을 구성하고 스프링 MVC를 활성화한다

6. 예제는 스프링 MVC 웹 애플리케이션이므로 리소스 핸들러는 클래스 패스의 루투를 기준으로 /static 위치에 있는 정적 콘텐츠를 제공하도록 등록된다(리소스 핸들러는 /public,/resource, /META-INF/resources의 정적 콘텐츠도 제공한다)

7.(웹스타터 의존성 덕분에 전이적으로) 톰캣이 클래스패스에 있으므로 8080 포트에서 수신대기 하는 내장 톰캣 컨테이너를 시작한다

여기서 핵심은 스프링 부트 자동 구성이 스프링을 구성하는 일을 하므로 여러분은 애플리케이션을 작성하는 일에만 집중해라

기존 자동 셋팅을 벗어나 다르게 구성하고 싶다면 어떻게 해야될까 그것은

자동 구성을 오버라이드하는 방법을 살펴볼것이다. 컨포넌트를 구성하는데 적용하는 방법도 알아보자구~

**구성을 사용자화하기(p81)**

스프링 부트 자동 구성 오버라이드하기

애플리케이션 보안설정하기

|  |
| --- |
| gradle  compile('org.springframework.boot:spring-boot-starter-security') |

스프링시큐리티를 애플리케이션에 자동 구성이일어나 가장 기본적인 설정을 만든다

이제 웹브라우저에서 애플리케이션을 열면 HTTP기본인증 대화상자가 뜬다

표준출력(stdout)으로 출력된 비밀번호를 입력한다

Using default security password : ~~~~~~

아이디는 user

사용자 정의 보안 구성하기

스프링 시큐리티의 자동 구성을 오버라이드할때는 WebSecurityConfigurerAdapter 를 확장해서 구성 클래스를 작성한다

|  |  |
| --- | --- |
| @Configuration  @EnableWebSecurity  **public** **class** SecurityConfig **extends** WebSecurityConfigurerAdapter {  @Autowired  **private** ReaderRepository readerRepository;  @Override  **protected** **void** configure(HttpSecurity http) **throws** Exception {  http  .authorizeRequests()  .antMatchers("/").access("hasRole('READER')")  .antMatchers("/\*\*").permitAll()  .and()  .formLogin()  .loginPage("/login")  .failureUrl("/login?error=true");  }  @Override  **protected** **void** configure(AuthenticationManagerBuilder auth) **throws** Exception {  auth.userDetailsService(**new** UserDetailsService() {  @Override  **public** UserDetails loadUserByUsername(String username)  **throws** UsernameNotFoundException {  **return** readerRepository.findOne(username);  }  });  }  } | |
| configure(HttpSecurity http) |  |
| configure(AuthenticationManagerBuilder auth) |  |

**viewController 처리**

**ArgumentResolver 처리**

|  |  |
| --- | --- |
| @SpringBootApplication  **public** **class** ReadingListApplication **extends** WebMvcConfigurerAdapter {  **public** **static** **void** main(String[] args) {  SpringApplication.run(ReadingListApplication.**class**, args);  }    @Override  **public** **void** addViewControllers(ViewControllerRegistry registry) {  registry.addViewController("/login").setViewName("login");  }    @Override  **public** **void** addArgumentResolvers(List<HandlerMethodArgumentResolver> argumentResolvers) {  argumentResolvers.add(**new** ReaderHandlerMethodArgumentResolver());  }    } | |
| addViewControllers(ViewControllerRegistry registry) | 뷰 맵핑 |
| addArgumentResolvers(List<HandlerMethodArgumentResolver> argumentResolvers) | 매개변수 처리 |

매개변수 처리 리졸버

|  |
| --- |
| **public** **class** ReaderHandlerMethodArgumentResolver **implements** HandlerMethodArgumentResolver {  @Override  **public** **boolean** supportsParameter(MethodParameter parameter) {  **return** Reader.**class**.isAssignableFrom(parameter.getParameterType());  }  @Override  **public** Object resolveArgument(MethodParameter parameter, ModelAndViewContainer mavContainer,  NativeWebRequest webRequest, WebDataBinderFactory binderFactory) **throws** Exception {  Authentication auth = (Authentication) webRequest.getUserPrincipal();  **return** auth != **null** && auth.getPrincipal() **instanceof** Reader ? auth.getPrincipal() : **null**;  }  } |

컨트롤러.

|  |
| --- |
| @Controller  @RequestMapping("/")  **public** **class** ReadingListController {  **private** ReadingListRepository readingListRepository;  **private** AmazonProperties amazonProperties;  @Autowired  **public** ReadingListController(ReadingListRepository readingListRepository, AmazonProperties amazonProperties) {  **this**.readingListRepository = readingListRepository;  **this**.amazonProperties = amazonProperties;  }    @RequestMapping(method = RequestMethod.***GET***)  **public** String readersBooks(Reader reader, Model model) {  List<Book> readingList = readingListRepository.findByReader(reader);  **if** (readingList != **null**) {  model.addAttribute("books", readingList);  model.addAttribute("reader", reader);  model.addAttribute("amazonID", amazonProperties.getAssociateId());  }  **return** "readingList";  }    @RequestMapping(method = RequestMethod.***POST***)  **public** String addToReadingList(Reader reader, Book book) {  book.setReader(reader);  readingListRepository.save(book);  **return** "redirect:/";  }  } |
| 시큐리티의 객체는 userDetail인터에페이스를..구현하면편하다  @Entity  **public** **class** Reader **implements** UserDetails {  **private** **static** **final** **long** ***serialVersionUID*** = 1L;  @Id  **private** String username;  **private** String fullname;  **private** String password;  **public** String getUsername() {  **return** username;  }  **public** **void** setUsername(String username) {  **this**.username = username;  }  **public** String getFullname() {  **return** fullname;  }  **public** **void** setFullname(String fullname) {  **this**.fullname = fullname;  }  **public** String getPassword() {  **return** password;  }  **public** **void** setPassword(String password) {  **this**.password = password;  }  // UserDetails 메서드  @Override  **public** Collection<? **extends** GrantedAuthority> getAuthorities() {  **return Arrays.*asList*(new SimpleGrantedAuthority("ROLE\_READER"));**  }  @Override  **public** **boolean** isAccountNonExpired() {  **return** **true**;  }  @Override  **public** **boolean** isAccountNonLocked() {  **return** **true**;  }  @Override  **public** **boolean** isCredentialsNonExpired() {  **return** **true**;  }  @Override  **public** **boolean** isEnabled() {  **return** **true**;  }  } |

**자동 구성에 숨은 기능 엿보기**

@ConditionalOnMissingBean **어노테이션 으로 자동 구성을 오버라이드 할수 있다 스프링 부트의 DataSourceAutoConfiguration에 정의된 JdbcTemplate빈 @ConditionalOnMissingBean의 작동을 설명하는 매우 간단한 예다**

|  |
| --- |
| **@Bean**  **@ConditionalOnMissionBean(JdbcOperations.class)**  **public JdbcTemplate jdbcTemplate(){**  **return new JdbcTemplate(this.dataSource);**  **}** |
| **jdbcTemplate() 는 @Bean 어노테이션이 붙어 있으며 필요할때 jdbcTemplate 빈을 구성한다**  **하지만 @ConditionalOnMission 어노테이션 붙어있어 JdbcTemplate 이 구현하는 인터패이스인 JdbcOperations 타입의 빈이 없을때만 작동한다**  **JdbcOperation타입의 빈을 이미 구성했다면 @ConditionalOnMission 어노테이션조건을 만족하지 못하므로 빈을 구성하지 않는다** |

SpringBootWebSecurityConfiguration (p93)

|  |
| --- |
| @Configuration  @EnableConfigurationProperties  @ConditionalOnClass({ EnableWebSecurity.**class**, AuthenticationEntryPoint.**class** })  @ConditionalOnMissingBean(WebSecurityConfiguration.**class**)  @ConditionalOnWebApplication  @EnableWebSecurity  **public** **class** SpringBootWebSecurityConfiguration {  **private** **static** List<String> *DEFAULT\_IGNORED* = Arrays.*asList*("/css/\*\*", "/js/\*\*",  "/images/\*\*", "/webjars/\*\*", "/\*\*/favicon.ico");  @Bean  @ConditionalOnMissingBean({ IgnoredPathsWebSecurityConfigurerAdapter.**class** })  **public** IgnoredPathsWebSecurityConfigurerAdapter ignoredPathsWebSecurityConfigurerAdapter() {  **return** **new** IgnoredPathsWebSecurityConfigurerAdapter();  }  **....** |

**프로퍼티를 이용하여 외부적으로 구성하기**

|  |
| --- |
| <http://docs.spring.io/spring-boot/docs/current/reference/html/common-application-properties.html>  # ===================================================================  # COMMON SPRING BOOT PROPERTIES  #  # This sample file is provided as a guideline. Do NOT copy it in its  # entirety to your own application. ^^^  # ===================================================================  # ----------------------------------------  # CORE PROPERTIES  # ----------------------------------------  # 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/v1.4.1.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/v1.4.1.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/v1.4.1.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.hazelcast.config= # The location of the configuration file to use to initialize Hazelcast.  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/v1.4.1.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/v1.4.1.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/v1.4.1.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/v1.4.1.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/v1.4.1.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/v1.4.1.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/v1.4.1.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/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/v1.4.1.RELEASE/spring-boot-actuator/src/main/java/org/springframework/boot/actuate/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 of [active profiles](http://docs.spring.io/spring-boot/docs/current/reference/html/howto-properties-and-configuration.html#howto-set-active-spring-profiles).  spring.profiles.include= # Unconditionally activate the specified comma separated profiles.  # SENDGRID ([SendGridAutoConfiguration](https://github.com/spring-projects/spring-boot/tree/v1.4.1.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/v1.4.1.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.max-http-post-size=0 # Maximum size in bytes of the HTTP post content.  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.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.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.suffix=.log # Log file name suffix.  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-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.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.worker-threads= # Number of worker threads.  # FREEMARKER ([FreeMarkerAutoConfiguration](https://github.com/spring-projects/spring-boot/tree/v1.4.1.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/v1.4.1.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/v1.4.1.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/v1.4.1.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.  # MULTIPART ([MultipartProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.1.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/v1.4.1.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.serialization-inclusion= # Controls the inclusion of properties during serialization. Configured with one of the values in Jackson's JsonInclude.Include enumeration.  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/v1.4.1.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 MOBILE DEVICE VIEWS ([DeviceDelegatingViewResolverAutoConfiguration](https://github.com/spring-projects/spring-boot/tree/v1.4.1.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/v1.4.1.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/v1.4.1.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/v1.4.1.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/v1.4.1.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/v1.4.1.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/session/SessionProperties.java))  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= # Flush mode for the Redis sessions.  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/v1.4.1.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/v1.4.1.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/v1.4.1.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/v1.4.1.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/v1.4.1.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.  # VELOCITY TEMPLATES ([VelocityAutoConfiguration](https://github.com/spring-projects/spring-boot/tree/v1.4.1.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/velocity/VelocityAutoConfiguration.java))  spring.velocity.allow-request-override=false # Set whether HttpServletRequest attributes are allowed to override (hide) controller generated model attributes of the same name.  spring.velocity.allow-session-override=false # Set whether HttpSession attributes are allowed to override (hide) controller generated model attributes of the same name.  spring.velocity.cache= # Enable template caching.  spring.velocity.charset=UTF-8 # Template encoding.  spring.velocity.check-template-location=true # Check that the templates location exists.  spring.velocity.content-type=text/html # Content-Type value.  spring.velocity.date-tool-attribute= # Name of the DateTool helper object to expose in the Velocity context of the view.  spring.velocity.enabled=true # Enable MVC view resolution for this technology.  spring.velocity.expose-request-attributes=false # Set whether all request attributes should be added to the model prior to merging with the template.  spring.velocity.expose-session-attributes=false # Set whether all HttpSession attributes should be added to the model prior to merging with the template.  spring.velocity.expose-spring-macro-helpers=true # Set whether to expose a RequestContext for use by Spring's macro library, under the name "springMacroRequestContext".  spring.velocity.number-tool-attribute= # Name of the NumberTool helper object to expose in the Velocity context of the view.  spring.velocity.prefer-file-system-access=true # Prefer file system access for template loading. File system access enables hot detection of template changes.  spring.velocity.prefix= # Prefix that gets prepended to view names when building a URL.  spring.velocity.properties.\*= # Additional velocity properties.  spring.velocity.request-context-attribute= # Name of the RequestContext attribute for all views.  spring.velocity.resource-loader-path=classpath:/templates/ # Template path.  spring.velocity.suffix=.vm # Suffix that gets appended to view names when building a URL.  spring.velocity.toolbox-config-location= # Velocity Toolbox config location. For instance `/WEB-INF/toolbox.xml`  spring.velocity.view-names= # White list of view names that can be resolved.  # SPRING WEB SERVICES ([WebServicesProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.1.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/v1.4.1.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-type=true # Enable "X-Content-Type-Options" header.  security.headers.frame=true # Enable "X-Frame-Options" header.  security.headers.hsts= # 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/v1.4.1.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/v1.4.1.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/security/oauth2/resource/ResourceServerProperties.java)  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/v1.4.1.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/v1.4.1.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/v1.4.1.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/v1.4.1.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/v1.4.1.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/v1.4.1.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= # 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/v1.4.1.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/v1.4.1.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.  # MONGODB ([MongoProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.1.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.  spring.data.mongodb.password= # Login password of the mongo server.  spring.data.mongodb.port=27017 # Mongo server port.  spring.data.mongodb.repositories.enabled=true # Enable Mongo repositories.  spring.data.mongodb.uri=mongodb://localhost/test # Mongo database URI. When set, host and port are ignored.  spring.data.mongodb.username= # Login user of the mongo server.  # DATA REDIS  spring.data.redis.repositories.enabled=true # Enable Redis repositories.  # NEO4J ([Neo4jProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.1.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.password= # Login password of the server.  spring.data.neo4j.repositories.enabled=true # Enable Neo4j repositories.  spring.data.neo4j.session.scope=singleton # Scope (lifetime) of the session.  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/v1.4.1.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.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/v1.4.1.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/v1.4.1.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/jdbc/DataSourceAutoConfiguration.java) & [DataSourceProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.1.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 reference.  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.dbcp.\*= # Commons DBCP specific settings  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.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 reference.  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/v1.4.1.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.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/v1.4.1.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/v1.4.1.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/v1.4.1.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/orm/jpa/JpaBaseConfiguration.java), [HibernateJpaAutoConfiguration](https://github.com/spring-projects/spring-boot/tree/v1.4.1.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/v1.4.1.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/v1.4.1.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= # 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/v1.4.1.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/v1.4.1.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.databaseDir= # Directory used for data storage.  spring.mongodb.embedded.storage.oplogSize= # Maximum size of the oplog in megabytes.  spring.mongodb.embedded.storage.replSetName= # 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/v1.4.1.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.host=localhost # Redis server host.  spring.redis.password= # Login password of the redis server.  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.  # ----------------------------------------  # INTEGRATION PROPERTIES  # ----------------------------------------  # ACTIVEMQ ([ActiveMQProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.1.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/v1.4.1.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/v1.4.1.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.  # HORNETQ ([HornetQProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.1.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/jms/hornetq/HornetQProperties.java))  spring.hornetq.embedded.cluster-password= # Cluster password. Randomly generated on startup by default.  spring.hornetq.embedded.data-directory= # Journal file directory. Not necessary if persistence is turned off.  spring.hornetq.embedded.enabled=true # Enable embedded mode if the HornetQ server APIs are available.  spring.hornetq.embedded.persistent=false # Enable persistent store.  spring.hornetq.embedded.queues= # Comma-separated list of queues to create on startup.  spring.hornetq.embedded.server-id= # Server id. By default, an auto-incremented counter is used.  spring.hornetq.embedded.topics= # Comma-separated list of topics to create on startup.  spring.hornetq.host=localhost # HornetQ broker host.  spring.hornetq.mode= # HornetQ deployment mode, auto-detected by default.  spring.hornetq.password= # Login password of the broker.  spring.hornetq.port=5445 # HornetQ broker port.  spring.hornetq.user= # Login user of the broker.  # JMS ([JmsProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.1.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.  # RABBIT ([RabbitProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.1.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.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/v1.4.1.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.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.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.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/v1.4.1.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/v1.4.1.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/v1.4.1.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/v1.4.1.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.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=ADMIN # 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 (previously health.\*)  management.health.db.enabled=true # Enable database 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.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, UNKNOWN, UP # Comma-separated list of health statuses in order of severity.  # INFO CONTRIBUTORS ([InfoContributorProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.1.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/v1.4.1.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=ADMIN # 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/v1.4.1.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/v1.4.1.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/v1.4.1.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/v1.4.1.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. |

|  |
| --- |
| **java -jar ...jar –spring.main.show-nanner=false** |

**YAML을 선호한다면 다음과 같이 application.yml로 YAML파일을 생성한다**

|  |
| --- |
| **spring:**  **main:**  **show-banner: false** |

**프로퍼티를 환경 변수로 설정할수도 있다**

**BASH나 zsh쉘을 사용한다면 export명령으로 프로퍼티를 설정한다**

|  |
| --- |
| **export spring\_main\_how\_banner=false**  **#마침표나 하이픈 대신 밑줄을 사용한다 \_** |

**여러 프로퍼티 소스로 프로퍼티를 사용할 수 있게한다**

1. 명령줄 인자

2. java:comp/env에서 얻을수 있는 JNDI속성

3. JVM 시스템 프로퍼티

4. 운영체제의 환경변수

5. random.\*로시작하는 프로퍼티 때문에 무작위로 생성된값(${random.long}처럼 다른 프로퍼티를 설정할때 참조)

6. 애플리케이션 외부에 있는 application.properties나 application.yml 파일

7. 애플리케이션 내부에 패키징된 application.properties나 application.yml파일

8. @PropertySource로 지정된 프로퍼티 소스

9. 기본 프로퍼티

**우선순위는 1번부터 이다. 1번이 아래순서들을 오버라이딩한다.**

**application.properties와 application.yml 파일은 다음 네곳에 배치할수 있다**

1. 외부적으로 애플리케이션이 작동하는 디렉토리 /config 하위 디렉토리
2. 외부적으로 애플리케이션이 작동하는 디렉토리
3. 내부적으로config 패키지
4. 내부적으로 클래스패스의 루트

동일한 레벨안에 properties와 yml이 있을시 yml있는 프로퍼티가 properties에있는 프로퍼티를 오버라이드한다

다른 레벨안에 있을시 안쪽 뎁스(/config)가 더 긴쪽이 상위 동일한 프로퍼티를 오버라이딩한다.

**탬플릿 캐싱 비활성화**

Thymeleaf 템플릿의 어떤 변경 사항도 반영하지 않았으니 Thymeleaf템플릿은 기본으로 캐시되어있다.

캐쉬되어있기때문에 즉시 반영이 어렵다 따라서 비활성화려면

spring.thymeleaf.cache=false 주면된다.

1. 프리마커 : spring.freemaker.cache

2. 그루비 템플릿 : spring.groovy.template.cache

3. 벨로시티 : spring.velocity.cache

**내장 서버 구성**

server.port=8080

**로깅 구성**

로깅 구성을 완전히 제어하려면 클래스패스 루트(src/main/resource)에 logback.xml파일을 생성해야한다.

대처하기

|  |
| --- |
| gradle 기존 로그 기본값 logback 제외하기  configurations{  all\*.exclude group:’org.springframework.boot’,module:’spring-boot-starter-logging’  } |
| 그리고 다른거 등록하기  compile(org.springframework.boot:spring-boot-starter-log4j’ //log4j2 |

**logback.xml**

<http://logback.qos.ch/manual/index.html>

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <configuration>  <appender name=*"STDOUT"* class=*"ch.qos.logback.core.ConsoleAppender"*>  <layout class=*"ch.qos.logback.classic.PatternLayout"*>  <Pattern>%d{HH:mm} %-5level %logger{36} - %msg%n</Pattern>  </layout>  </appender>    <logger name=*"example.logback.level.grandparents"* level=*"TRACE"*/>  <logger name=*"example.logback.level.grandparents.parents.children"* level=*"INFO"*/>  <root level=*"DEBUG"*>  <appender-ref ref=*"STDOUT"* />  </root>  </configuration> |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <configuration>  <include resource=*"org/springframework/boot/logging/logback/base.xml"*/>  <!-- log4jdbc-log4j2 -->  <logger name=*"jdbc.sqlonly"* level=*"ERROR"*/>  <logger name=*"jdbc.sqltiming"* level=*"INFO"*/>  <logger name=*"jdbc.audit"* level=*"INFO"*/>  <logger name=*"jdbc.resultset"* level=*"ERROR"*/>  <logger name=*"jdbc.resultsettable"* level=*"ERROR"*/>  <logger name=*"jdbc.connection"* level=*"INFO"*/>  <logger name=*"com.libqa.web"* level=*"DEBUG"*/>  </configuration> |

**logback.xml을 생성하지 않고도 설정할수 있다.**

|  |
| --- |
| logging:  path: /var/logs/  file: log.log  level:  root: WARN  org:  pringframework:  security: DEBUG |
| **로그항목은 10MB될때마자 교체된다** |

**logback.xml이 아니라 다른걸로 이름변경하고싶다면**

|  |
| --- |
| logging:  config:  classpath: loging-config.xml |

**데이터 소스 구성**

개발중에는 H2내장 데이터 베이스정도면 가능하나. 영구적인 데이터베이스 솔루션을 고려하고자할땐.

명시적으로 자체 DataSource빈을 구성할 수도 있지만, 보통 이과정이 필요하다.

|  |
| --- |
| spring:  datasource:  [url: jdbc:mysql://localhost/redinglist](url:jdbc:mysql://localhost/redinglist)  username: name  password: pwd  driver-class-name: com.mysql.jdbc.Driver |

스프링 부투는 DataSource 빈을 자동 구성할 때 커넥션 정보를 사용한다 톰캣의 풀링 DataSource가 클래스 패스에 있다면 DataSource빈은 풀링될것이다 아니면 다른 커넥션 풀 구현체를 찾아 사용할것이다

HikariCP

CommonsDBCP

CommonsDBCP2

원하는 풀링 으로 변경가능하다

|  |
| --- |
| spring:  datasource:  jndi-name: java:/comp/env/jdbc/redds |

**외부에서 애플리케이션 빈 구성하기**

**Thymeleaf 템플릿을 바꿔서 구현할수있다 아마존 구현**

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>Reading List</title>  <link rel=*"stylesheet"* th:href=*"@{/css/style.css}"*></link>  </head>  <body>  <form method=*"POST"* th:action=*"@{/logout}"*>  <input type=*"submit"* value=*"Logout"* />  </form>    <h2><span th:text=*"${reader.fullname}"*>Name</span>'s Reading List</h2>  <div th:unless=*"${#lists.isEmpty(books)}"*>  <dl th:each=*"book : ${books}"*>  <dt class=*"bookHeadline"*>  <a th:href=*"'http://www.amazon.com/gp/product/'*  *+ ${book.isbn}*  *+ '/tag=' + ${amazonID}"*  th:text=*"${book.title}"*>Title</a> by  <span th:text=*"${book.author}"*>Author</span>  (ISBN: <span th:text=*"${book.isbn}"*>ISBN</span>)  </dt>  <dd class=*"bookDescription"*>  <span th:if=*"${book.description}"*  th:text=*"${book.description}"*>Description</span>  <span th:if=*"${book.description eq null}"*>  No description available</span>  </dd>  </dl>  </div>  <div th:if=*"${#lists.isEmpty(books)}"*>  <p>You have no books in your book list</p>  </div>  <hr/>    <h3>Add a book</h3>  <form method=*"POST"* th:action=*"@{/}"*>  <label for=*"title"*>Title:</label>  <input type=*"text"* name=*"title"* size=*"50"*></input><br />  <label for=*"author"*>Author:</label>  <input type=*"text"* name=*"author"* size=*"50"*></input><br />  <label for=*"isbn"*>ISBN:</label>  <input type=*"text"* name=*"isbn"* size=*"15"*></input><br />  <label for=*"description"*>Description:</label><br />  <textarea name=*"description"* cols=*"80"* rows=*"5"*></textarea><br />  <input type=*"submit"* value=*"Add Book"* />  </form>  </body>  </html> |

**위내용에 필요한 amazonid가 필요하다 컨트롤러에서 넘겨주자**

|  |
| --- |
| @Controller  @RequestMapping("/")  @ConfigurationProperties("prefix=”amazon")  **public** **class** ReadingListController {  **private** ReadingListRepository readingListRepository;  **private** String associateId;  @Autowired  **public** ReadingListController(ReadingListRepository readingListRepository, AmazonProperties amazonProperties) {  **this**.readingListRepository = readingListRepository;  **this**.amazonProperties = amazonProperties;  }  **public void setAssociateId(String associateId){**  **this.associateId = associateId;**  **}**  @RequestMapping(method = RequestMethod.***GET***)  **public** String readersBooks(Reader reader, Model model) {  List<Book> readingList = readingListRepository.findByReader(reader);  **if** (readingList != **null**) {  model.addAttribute("books", readingList);  model.addAttribute("reader", reader);  model.addAttribute("amazonID", associateId);  }  **return** "readingList";  }    @RequestMapping(method = RequestMethod.***POST***)  **public** String addToReadingList(Reader reader, Book book) {  book.setReader(reader);  readingListRepository.save(book);  **return** "redirect:/";  }  } |
| amazon.associateId=asdasdasdvalue  amazon:  associateId : hhhhhhvalue |
| @ConfigurationProperties 으로 프로퍼티를 가져올수있다. |

**클래스 하나에 프로퍼티 모으기**

|  |
| --- |
| @Component  @ConfigurationProperties("amazon")  **public** **class** AmazonProperties {  **private** String associateId;  **public** **void** setAssociateId(String associateId) {  **this**.associateId = associateId;  }  **public** String getAssociateId() {  **return** associateId;  }  } |
| @Controller  @RequestMapping("/")  **public** **class** ReadingListController {  **private** ReadingListRepository readingListRepository;  **private** AmazonProperties amazonProperties;  @Autowired  **public** ReadingListController(ReadingListRepository readingListRepository, AmazonProperties amazonProperties) {  **this**.readingListRepository = readingListRepository;  **this**.amazonProperties = amazonProperties;  }    @RequestMapping(method = RequestMethod.***GET***)  **public** String readersBooks(Reader reader, Model model) {  List<Book> readingList = readingListRepository.findByReader(reader);  **if** (readingList != **null**) {  model.addAttribute("books", readingList);  model.addAttribute("reader", reader);  model.addAttribute("amazonID", amazonProperties.getAssociateId());  }  **return** "readingList";  } |

**프로파일 구성**

런타임 배포할때 몇몇 세부 구성은 다를수 있다.

**@Profile** 어노테이션은 production 프로파일을 런타임에서 활성화했을 때만 해당 구성을 적용한다 production 프로파일을 활성되지 않을 때는 이 설정을 무시하고, 오버라이드된 보안 구성이 없으므로 자동으로 구성되 보안 구성을 제공할 것이다.

|  |
| --- |
| spring:  profiles:  active: production |

하지만 스프링 부투는 많은 구성을 자동으로 구성하므로 단순히 @Profile어노ㅔ이션을 붙이려고 명시적으로 구성을 작성하는 것은 오히려 불편할수도 있다. 스프링부트는 application.properties와 application.yml설정한 프로퍼티에도 프로파일을 지원한다.

**프로파일에 특화된 프로퍼티 파일 다루기**

application.properties로 구성 프로퍼티를 설정한다면 application-{profile}.properties형태의 추가적인 프로퍼티 파일을 생성하여 프로파일에 특화된 프로퍼티를 제공할수 있다.

앞에서 설명한 로깅 시나로오대로 하면 개발용 구성은 application-development.properties파일에 들어가고 콘설에 상세한 로그를 출력하도록 다음과 같이 설정할수도 있겠다

**YAML파일로 여러 프로파일 구성**

application-{profile}.yml파일을 생성하고 프로파일되지 않은 프로퍼티는 application.yml에 넣으면된다.

|  |
| --- |
| amazon:  associateId: habuma-20  logging:  level:  root: INFO  ---  spring:  profiles: development  logging:  level:  root: DEBUG  ---  spring:  profiles: production  logging:  level:  root: DEBUG |
| 하나의 파일 application.yml 파일은 ---하이픈3개로 나누었다  spring.profile를 development로 설정하며 development 프로파일이 적용된다.  production으로 설정했으므로 production파일이 활성화 되었을때 적용된다.  명시하지 않은 profile값은. 공통으로 사용학나 활성화된 프로퍼티를 설정하지 않을을떄 기본값으로 설정된다. |

**어플리케이션 오류 페이지 사용자 정의하기**

스프링부트는 화이트 라벨 오류 페이지를 기본제공한다

스프링 부트가 자동으로 구성한 기본 오류 핸들러는 error 뷰를 찾는다 찾을수 없으면 기본 화이트라벨 뷰를 사용한다

결국 사용자 정의할 뷰는 다음과같이 오류뷰룰 해석할때 사용될 뷰 리졸버에 따라 달라진다.

스프링의 View 인터페이스를 구현하며 ID가 error인 빈(스프링의 BeanNameViewResolver가 해석)

Thymeleaf를 사용하면 Thymeleaf템플릿 error.html

FreeMarker를 사용한다면 FreeMarker 템플릿 error.ftl

Velocity를 사용한다면 Velocity 템플릿 error.vm

jsp를 사용한다면 jsp템플릿 error.jsp

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>Oops!</title>  <link rel=*"stylesheet"* th:href=*"@{/css/style.css}"*></link>  </head>  <body>  <div class=*"errorPage"*>  <span class=*"oops"*>Oops!</span><br />  <img th:src=*"@{/images/MissingPage.png}"*></img>  <p>There seems to be a problem with the page you requested  (<span th:text=*"${path}"*></span>).</p>  <p th:text=*"${'Details: ' + message}"*></p>  </div>  </body>  </html> |
| 이사용자 오류 템플릿을 error.html로 저장하고 Thymeleaf템플릿 리졸버가 찾을 수 있도록 템플릿 디렉터리에 둬야한다. 템플릿이 클래스패스의 루트에 있도록 src/main/resource/templates 디렉터리에 |
|  |
| **body** {  background-color: *#cccccc*;  font-family: *arial,* *helvetica,* *sans-serif*;  }  *.bookHeadline* {  font-size: *12pt*;  font-weight: *bold*;  }  *.bookDescription* {  font-size: *10pt*;  }  **label** {  font-weight: *bold*;  }  *.error* {  color: *red*;  }  *.errorPage* {  text-align: *center*;  }  *.oops* {  font-size: *76pt*;  } |

대체로 오류 페이지는 오류 메시지와 이미지를 보여주는 간단한 템프릿이다.

기본적으로 스프링 부트에서는 오류뷰에 다음오류 속성을 표시할수 있도록 제공한다

|  |  |
| --- | --- |
| timestamp | 오류발생시각 |
| status | HTTP상태코드 |
| error | 오류발생이유 |
| exception | 예외클래스이름 |
| message | 예외 메시지 (예외 때문에 발생한 오류일때) |
| errors | BindingResult 예외로 발생한 모든 오류(예외 때문에 발생한 오류일때) |
| trace | 예외 스택 트레이스(예외 때문에 발생한 오류일때) |
| path | 오류가 발생했을 때 요청한 URL경로. |

어플리케이션 이 실행중일때 이미지가 나타나게하려면 resource/static/images나 resoutce/public/images에 두어야한다

스프링 부트 테스트하기

통합 테스트를 위한 자동 구성(p119)

SpringJUnit4ClassRunner를 이용한 테스트

|  |
| --- |
| @RunWith(SpringJUnit4ClassRunner.**class**)  @~~SpringApplicationConfiguration~~ (~~classes~~ = ReadingListApplication.**class**)  @WebAppConfiguration  **public** **class** MockMvcWebTests {  @Autowired  **private** WebApplicationContext webContext;  **private** MockMvc mockMvc;  @Before  **public** **void** setupMockMvc() {  mockMvc = MockMvcBuilders  .*webAppContextSetup*(webContext)  .apply(springSecurity())  .build();  }  @Test  **public** **void** homePage\_unauthenticatedUser() **throws** Exception {  mockMvc.perform(*get*("/"))  .andExpect(*status*().is3xxRedirection())  .andExpect(*header*().string("Location", "http://localhost/login"));  }  @Test  @WithUserDetails("craig")  **public** **void** homePage\_authenticatedUser() **throws** Exception {  Reader expectedReader = **new** Reader();  expectedReader.setUsername("craig");  expectedReader.setPassword("password");  expectedReader.setFullname("Craig Walls");  mockMvc.perform(*get*("/"))  .andExpect(*status*().isOk())  .andExpect(*view*().name("readingList"))  .andExpect(*model*().attribute("reader", *samePropertyValuesAs*(expectedReader)))  .andExpect(*model*().attribute("books", *hasSize*(0)))  .andExpect(*model*().attribute("amazonID", "habuma-20"));  }  } |
| @RunWith 어노테이션에는 SpringJUnit4ClassRunner.class를 전달항스프링 통합 테스트를 활성화한다  @Configuration 어노테이션에는 컨텍스트를 어떻게 로드할지 지정한다. |

스프링MVC모킹하기 (p122)

웹보안테스트하기(p128)

실행중인 어플리케이션 테스트하기(p133)

임의의 포트로 서버 실행하기(p135)

셀레늄으로 html페이지 테스트하기(p137)

그루비로 시작하기(p147)

@Grab어노테이션으로 의존성 관리하기(p153)

배포가능한 아티팩트 생성하기(p160)

스프링부트에 그레일즈 적용하기(p164)

그레일즈3와 스프링 부트 함께 사용하기(p177)

액추에이터로 내부 들여다보기(p190)

엑추에디터 원격 쉘에 접속하기(p211)

스프링부터 어플리케이션 배포하기(p237);

WAR파일생성하기

|  |
| --- |
| apply plugin:’war’  war{  baseName=’ff’  version=’0.0.0.1SNAPSHOT’  } |

출시용 프로파일 생성하기

|  |
| --- |
| @Bean  @Profile(“production”)  public DataSource dataSource(){  ...  } |

데이터베이스 마이그레이션 활성화하기(p247)

|  |
| --- |
| spring:  jpa:  hibernate:  ddl-auto: create-drop |
| 이설정은 어플리케이션이 재시작할때마다 DB스키마를 완전 초기화한후 다시생성하므로 출시 환경에 적합하지 않다 |
|  |
| spring:  jpa:  hibernate:  ddl-auto: none |
| 하이버네이트가 테이블을 생성하지 못하도록 함 |

그대신 schema.sql에 스키마를 정의할 수 있다. schema.sql에 정의된 스크립트는 처음에는 잘 작동하지만 이후에 애플리케이션을 실행할때는 이미 대상 테이블이 있어 초기화 스크립트가 작동하지 않을것이다 따라서 초기 스크립트를 작성할때는 이미 수행한 작업을 반복하지 않도록 세슴 주의기울어야 한다

**Flyway로 데이터베이스 마이그레이셔 정의**

compile(‘org.flywaydb:flyway-core’)

**Liquibase로 데이터 베이스 마이그레이션 정의**

compile(‘org.liquibase:liquibase-core’)

|  |
| --- |
| databaseChangeLog:  - changeSet:  id: 1  author: habuma  changes:  - createTable:  tableName: reader  columns:  - column:  name: username  type: varchar(25)  constraints:  unique: **true**  nullable: **false**  - column:  name: password  type: varchar(25)  constraints:  nullable: **false**  - column:  name: fullname  type: varchar(50)  constraints:  nullable: **false**  - createTable:  tableName: book  columns:  - column:  name: id  type: bigserial  autoIncrement: **true**  constraints:  primaryKey: **true**  nullable: **false**  - column:  name: author  type: varchar(50)  constraints:  nullable: **false**  - column:  name: description  type: varchar(1000)  constraints:  nullable: **false**  - column:  name: isbn  type: varchar(10)  constraints:  nullable: **false**  - column:  name: title  type: varchar(250)  constraints:  nullable: **false**  - column:  name: reader\_username  type: varchar(25)  constraints:  nullable: **false**  references: reader(username)  foreignKeyName: fk\_reader\_username  - createSequence:  sequenceName: hibernate\_sequence  - insert:  tableName: reader  columns:  - column:  name: username  value: craig  - column:  name: password  value: password  - column:  name: fullname  value: Craig Walls |
| 데이터베이스 초기화 Liquibase |

Resource loader

리소스를 로딩할때.

파일

|  |
| --- |
| Resource resource = resourceLoader.getResource("classpath:hibernate/asd.hbm.xml"); factory.setMappingLocations(resource); |

ant pattren

http://stackoverflow.com/questions/25405167/finding-resources-with-pathmatchingresourcepatternresolver-and-urlclassloader-in

|  |
| --- |
| ClassLoader cl = this.getClass().getClassLoader(); ResourcePatternResolver resolver = new PathMatchingResourcePatternResolver(cl); Resource[] resources = resolver.getResources("classpath\*:/\*.xml") ; for (Resource resource: resources){  logger.info(resource.getFilename()); } |