Springboot 入门

一.spring java配置方式

> Java配置是Spring4.x推荐的配置方式，用于替代xml配置。

1.1. @Configuration 和 @Bean

Spring的Java配置方式是通过 @Configuration和 @Bean 这两个注解实现的：

> 1、@Configuration 作用于类上，相当于一个xml配置文件；

> 2、@Bean 作用于方法上，相当于xml配置中的；

1.2示例代码

1.2.1 pom配置文件

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>xin.chenjunbo</groupId>
    <artifactId>springjavaconfig</artifactId>
    <version>1.0-SNAPSHOT</version>
    <dependencies>

        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-webmvc</artifactId>
            <version>4.3.10.RELEASE</version>
        </dependency>
        <dependency>
            <groupId>com.mchange</groupId>
            <artifactId>c3p0</artifactId>
            <version>0.9.5.2</version>
        </dependency>
    </dependencies>
    <build>
        <pluginManagement>
            <plugins>
                <!-- 配置Tomcat插件 -->
```

```
                <plugin>
                    <groupId>org.apache.tomcat.maven</groupId>
                    <artifactId>tomcat7-maven-plugin</artifactId>
                    <version>2.2</version>
                </plugin>
            </plugins>
        </pluginManagement>
    </build>
</project>
```

1.2.2 pojo对象

```
/**
 * Created by jackiechan on 2017/12/18/下午10:12.
 */
public class User {
    private String uid;
    private String userName;
    private int age;

    public String getUid() {
        return uid;
    }

    public void setUid(String uid) {
        this.uid = uid;
    }

    public String getUserName() {
        return userName;
    }

    public void setUserName(String userName) {
        this.userName = userName;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    @Override
    public String toString() {
```

```
        return "User{" +
                "uid='" + uid + '\'' +
                ", userName='" + userName + '\'' +
                ", age=" + age +
                '}';
    }
}
```

1.2.3 dao 对象

```
/**
 * Created by jackiechan on 2017/12/18/下午10:12.
 此处没有使用注解,目的是为了演示 bean 注解
 */
public class UserDao {

    public List<User> findAllUser() {
        List<User> list = new ArrayList<>();
        for (int i = 0; i < 10; i++) {
            User user = new User();
            user.setAge(10 + i);
            user.setUid("uid" + i);
            user.setUserName("zhangsan" + i);
            list.add(user);
        }
        return list;
    }
}
```

1.2.4 service 对象

```
/**
 * Created by jackiechan on 2017/12/18/下午10:12.
 */
@Service
public class UserService {
    @Autowired
    private UserDao userDao;

    public List<User> findAllUser() {
        return userDao.findAllUser();
    }
}
```

1.2.5 springConfig类

```
/**
 * Created by jackiechan on 2017/12/18/下午10:16.
 */
@Configuration //声明当前类是 spring 的配置文件xml,我们可以将各种配置放到这里,比如创建对象等
@ComponentScan(basePackages = "xin.chenjunbo.springbootdemodelete")// 包扫描,会自动帮我们扫描注解
public class SpringConfig {
  /**
  创建 userdao对象,相当于配置文件中的 bean 标签,当这个配置文件被加载时候,这个注解会被解析,会调用此方法创建对象,实际开发中,我们自己写的对象都是通过 commpent 相关注解创建的,此注解主要用于创建我们无法添加注解的引入依赖类
  */
    @Bean
    public UserDao getUserDao() {
        return new UserDao();
    }
}
```

1.2.6 测试类

```
public class Test {

    public static void main(String[] args) {
        //SpringApplication.run(SpringbootdemodeleteApplication.class, args);
        AnnotationConfigApplicationContext configApplicationContext = new
AnnotationConfigApplicationContext(SpringConfig.class);//注意此处使用的不再是 xml
context,而是AnnotationConfigApplicationContext
        UserService userService =
configApplicationContext.getBean(UserService.class);//获取对象
        List<User> allUser = userService.findAllUser();//调用方法
        System.out.println(allUser);
    }
}
```

1.3 加载其他配置文件

1.3.1 加载 properties 文件

```
/**
 * Created by jackiechan on 2017/12/18/下午10:16.
 */
@Configuration //声明当前类是 spring 的配置文件xml,我们可以将各种配置放到这里,比如创建
对象等
@ComponentScan(basePackages = "xin.chenjunbo.springbootdemodelete")// 包扫描,会自动
帮我们扫描注解
@PropertySource(value= {"classpath:jdbc.properties"})//通过@PropertySource可以指定
读取的配置文件，通过@Value注解获取值
public class SpringConfig {
    @Value("${jdbc.url}")
    private String jdbcUrl;

  /**
  创建 userdao对象,相当于配置文件中的 bean 标签
  */
    @Bean
    public UserDao getUserDao() {
        return new UserDao();
    }
}
```

> 可以引入多个 properties 文件,因为注解中的 value 是个数组,可以追加多个值
> @PropertySource(value=
> {"classpath:jdbc.properties","classpath:jdbc111.properties","classpath:jdbc222.properties"
> })

如果对应的配置文件不存在可能会出错,可以在注解后面追加@PropertySource(value = {"classpath:jdbc.properties"},ignoreResourceNotFound = true) ,设置为 true 即可

1.3.2 配置数据库连接

```
/**
 * Created by jackiechan on 2017/12/25/下午10:34
 */
@Configuration
public class SpringConfig {
    @Value("${jdbc.url}")
    private String jdbcUrl;
    @Value("${jdbc.username}")
    private String username;
    @Value("${jdbc.password}")
    private String password;
    @Value("${jdbc.className}")
    private String className;

    @Bean
    public DataSource dataSource() throws PropertyVetoException {
        ComboPooledDataSource comboPooledDataSource = new ComboPooledDataSource();
        comboPooledDataSource.setJdbcUrl(jdbcUrl);
        comboPooledDataSource.setUser(username);
        comboPooledDataSource.setPassword(password);
        comboPooledDataSource.setDriverClass(className);
        //其他属性此处忽略
        return comboPooledDataSource;
    }
}
```

1.3.3 加载 xml 配置文件

```
在特殊情况下,我们必须使用 xml 文件,因此需要导入 xml 配置文件
@ImportResource(value = {"xml1","xml2"}) 只需要将每个 xml 文件添加进来即可,具体到文件名
```

二. SpringBoot

2.1 什么是 springboot

Spring Boot是由Pivotal团队提供的全新框架，其设计目的是用来简化新Spring应用的初始搭建以及开发过程。该框架使用了特定的方式来进行配置，从而使开发人员不再需要定义样板化的配置。通过这种方式，Spring Boot致力于在蓬勃发展的快速应用开发领域（rapid application development）成为领导者。

spring大家都知道，boot是启动的意思。所以，spring boot其实就是一个启动spring项目的一个工具而已。从最根本上来讲，Spring Boot就是一些库的集合，它能够被任意项目的构建系统所使用。
以前在写spring项目的时候，要配置各种xml文件，还记得曾经被ssh框架支配的恐惧。随着spring3，spring4的相继推出，约定大于配置逐渐成为了开发者的共识，大家也渐渐的从写xml转为写各种注解，在spring4的项目里，你甚至可以一行xml都不写。

虽然spring4已经可以做到无xml，但写一个大项目需要茫茫多的包，maven配置要写几百行，也是一件很可怕的事。

现在，快速开发一个网站的平台层出不穷，nodejs，php等虎视眈眈，并且脚本语言渐渐流行了起来（Node JS，Ruby，Groovy，Scala等），spring的开发模式越来越显得笨重。

在这种环境下，spring boot伴随着spring4一起出现了。
springboot 的使用很简单,我们只需要将原先我们的 xml 配置中的内容通过 java 方式配置过去即可,大部分配置已经被 springboot 自己装配,我们只需要将需要我们自己写的配置单独写出来即可

2.2 hello world

2.2.1 添加依赖

> springboot 的依赖添加很简单,只要将项目的 parent 指定为 springboot 即可

```
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>1.5.9.RELEASE</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
```

2.2.2 导入 springboot 的 web 依赖

```
<!--此处不需要添加版本,由 parent 统一管理-->
    <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
```

2.2.3 添加 springboot 插件

```
    <plugin>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
```

2.3.4 入门类

```
/**
 * Created by jackiechan on 2017/12/25/下午10:55
 * 当前类既是一个 controller 又是一个配置文件,又是一个启动文件
 */
@SpringBootApplication
@Configuration
@Controller
public class HelloWorldApplication {
    @RequestMapping("helloworld")
    @ResponseBody
    public  String helloWorld() {
        return "hello moto";
    }
}
```

> 1、@SpringBootApplication：Spring Boot项目的核心注解，主要目的是开启自动配置。可以将程序以 web 方式运行
>
> 2、@Configuration：这是一个配置Spring的配置类；
>
> 3、@Controller：标明这是一个SpringMVC的Controller控制器；

2.3.5 测试类

```
/**
 * Created by jackiechan on 2017/12/25/下午11:00
 */
public class Test {
    public static void main(String[] args) {
        SpringApplication.run(HelloWorldApplication.class, args);//启动 spingboot
项目
    }
}
```

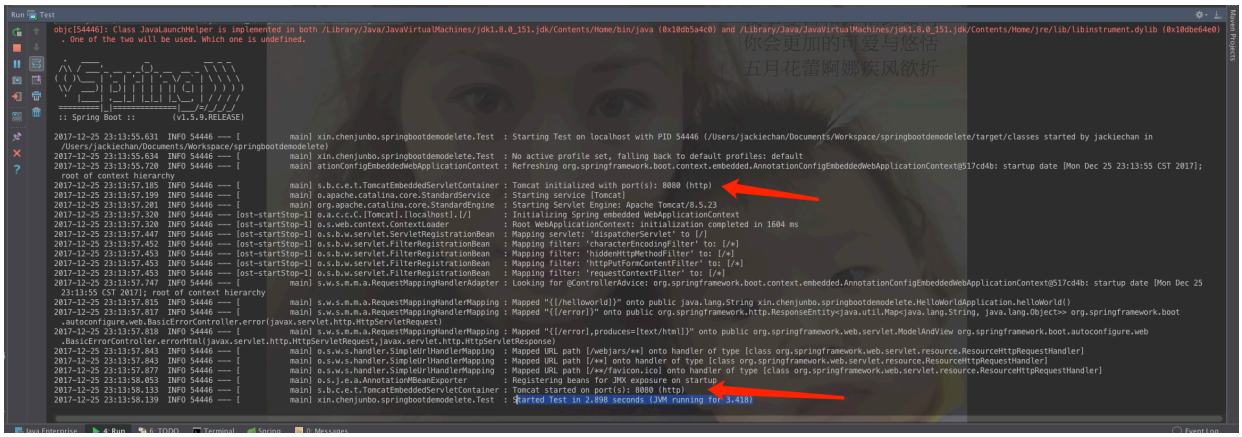
2.3.6 启动方式1

```
直接运行 Test 类的 main 方法即可
```

2.3.7启动方式2

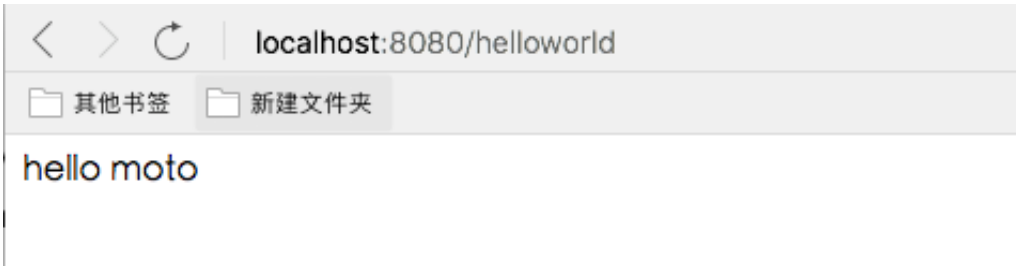
在配置了 springboot 的插件后,可以使用 maven 指令启动项目 指令为 spring-boot:run

2.3.8 效果



2.3.9访问

访问 http://localhost:8080/helloworld



三. Springboot 相关核心内容

3.1 入口类

springboot 一般有一个* application 结尾的类作为入口类,内部一个 main 方法,是一个标准的 java 程序

3.2 常见注解

3.2.1 @SpringBootApplication

@SpringBootApplication注解是Spring Boot的核心注解,用于标注程序是一个springboot 程序,它是一个组合注解,由多个注解组合而成

```
@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
@Documented
@Inherited
@SpringBootConfiguration
@EnableAutoConfiguration
@ComponentScan(excludeFilters = {
        @Filter(type = FilterType.CUSTOM, classes = TypeExcludeFilter.class),
        @Filter(type = FilterType.CUSTOM, classes =
AutoConfigurationExcludeFilter.class) })
public @interface SpringBootApplication {
  //此处忽略接口内部内容
}
```

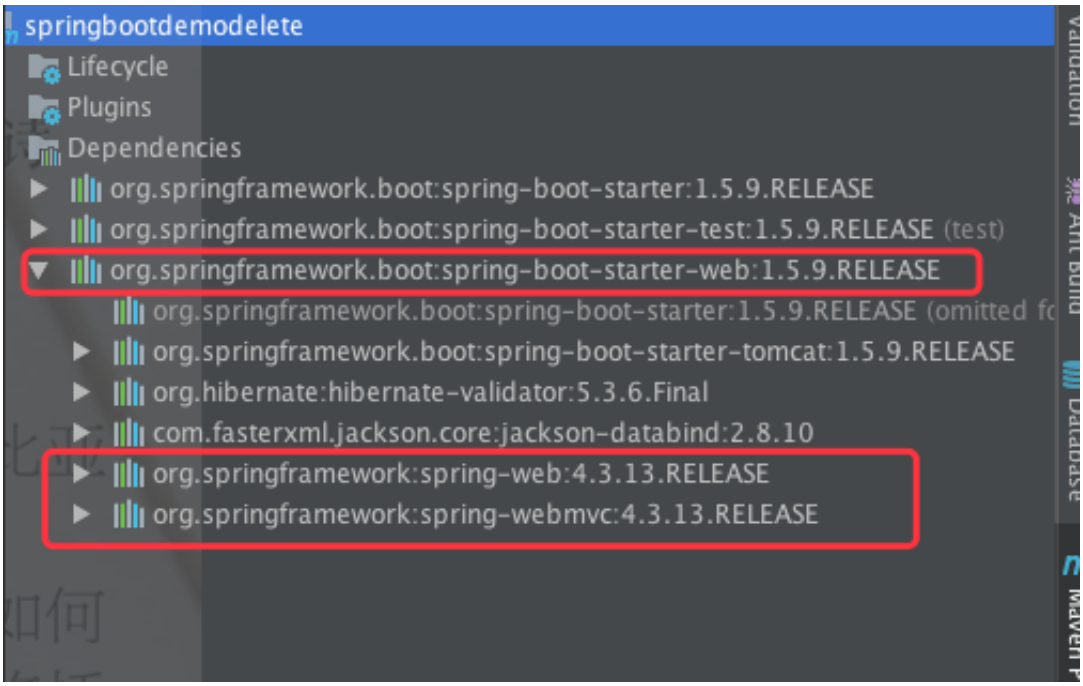
3.2.2 @SpringBootConfiguration注解

在@SpringBootApplication注解包括了一个@SpringBootConfiguration注解 它其实也是一个组合注解
在Spring Boot项目中推荐使用@SpringBootConfiguration替代@Configuration,因为@SpringBootConfiguration包含了@Configuration注解

```
@Target({ElementType.TYPE})
@Retention(RetentionPolicy.RUNTIME)
@Documented
@Configuration
public @interface SpringBootConfiguration {
}
```

3.2.3 @EnableAutoConfiguration注解

启用自动配置，该注解会使Spring Boot根据项目中依赖的jar包自动配置项目的配置项,这也是springboot 的核心注解之一,我们只需要将项目需要的依赖包假如进来,它会自动帮我们配置这个依赖需要的基本配置
比如我们的项目引入了spring-boot-starter-web依赖,springboot 会自动帮我们配置 tomcat 和springmvc



3.2.4 @ComponentScan 注解

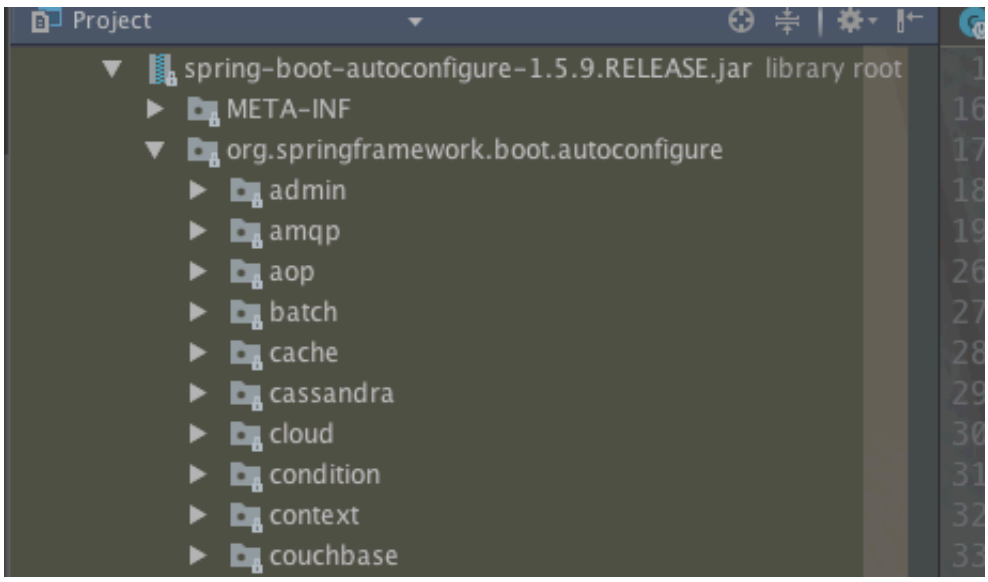
默认扫描@SpringBootApplication类所在包的同级目录以及它的子目录。

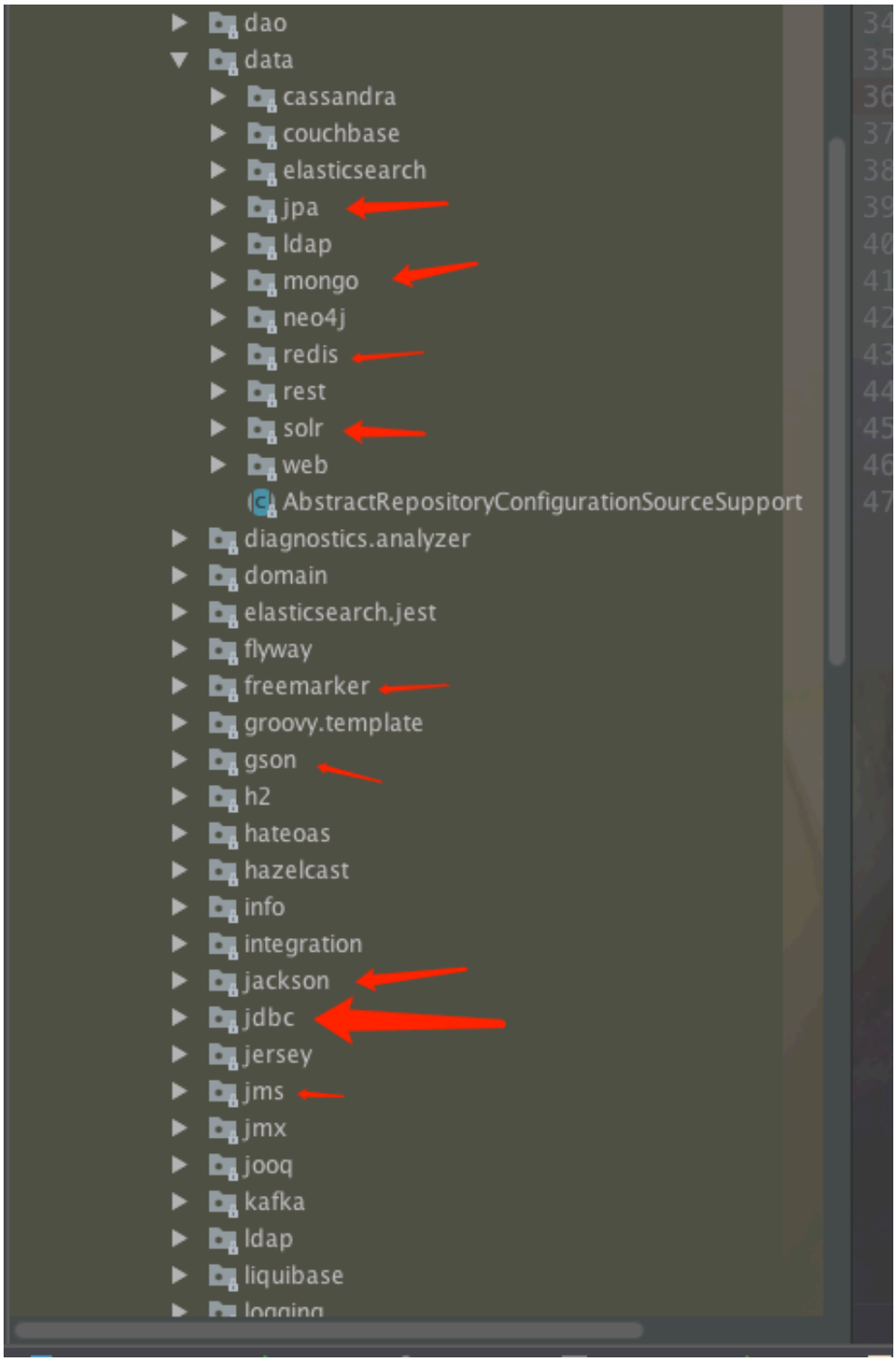
3.2.5 关闭自动配置

springboot 的自动配置可以帮我们节省很多时间,但是有时候如果我们不想在引入依赖包的情况自动配置,则可以通过相关设置取消

3.2.5.1 springboot 支持的自动配置

在spring-boot-autoconfigure包中包含了支持的自动装配依赖,截图只是其中一部分,我们开发中常见的依赖都做了基本配置





3.2.5.2 设置不自动装配

在@SpringBootApplication(exclude = {JpaRepositoriesAutoConfiguration.class, RedisAutoConfiguration.class})注解内部将不需要自动配置的依赖通过exclude参数指定即可,可以指定多个类

3.3 条件注解Condition Annotations

条件注解用于设置当前配置文件的加载条件,比如在某些情况下才会加载
按照使用情况,分为以下几种:
1. 类条件注解
2. 对象条件注解
3. 属性条件注解
4. 资源条件注解
5. web 程序注解
6. spel 表达式条件注解

3.3.1 类条件注解

3.3.1.1 @ConditionalOnClass

此注解的作用是指定当前配置必须在指定类存在的情况下才会触发
比如 redis 的自动配置

```
@Configuration
@ConditionalOnClass({ JedisConnection.class, RedisOperations.class, Jedis.class
})//指定必须在存在相关类的情况下才会自动配置 redis, 因为不存在这些文件说明你没有引用
redis, 也就没必要配置了
@EnableConfigurationProperties(RedisProperties.class)
  public class RedisAutoConfiguration {

  }
```

3.3.1.2 ConditionalOnMissingClass

此注解用于指定必须在缺少某个类的情况下才会生效,classpath中不存在该类时起效

3.3.2 对象类型注解

3.3.2.1@ConditionalOnBean

DI容器中存在该类型Bean时起效

3.3.2.2 @ConditionalOnMissingBean

DI容器中不存在该类型Bean时起效

3.3.2.3 @ConditionalOnSingleCandidate

DI容器中该类型Bean只有一个或@Primary的只有一个时起效

3.3.3 属性注解

3.3.3.1 @ConditionalOnProperty

参数设置或者值一致时起效

3.3.4 spel 表达式注解

3.3.4.1 @ConditionalOnExpression

SpEL表达式结果为true时

3.3.5 资源注解

3.3.5.1 @ConditionalOnResource

指定的文件存在时起效

3.3.6 web 应用注解

3.3.6.1 @ConditionalOnWebApplication

Web应用环境下起效

3.3.6.2 @ConditionalOnNotWebApplication

非Web应用环境下起效

3.3.7 其他注解

3.3.7.1 @ConditionalOnJndi

指定的JNDI存在时起效

3.3.7.2 @ConditionalOnJava

指定的Java版本存在时起效

3.4 Springboot 全局配置文件

```
#Spring Boot项目使用一个全局的配置文件application.properties或者是application.yml，在resources目录下或者类路径下的/config下，一般我们放到resources下,我们可以通过这个配置来修改我们项目中需要自定义的内容,建议使用 yml 格式的配置文件
#修改 tomcat 端口号
server.port=80
#修改 springmvc dispatcherservlet 拦截后缀 默认是/
server.sevlet-path=*.do
#设置控制台日志级别
logging.level.org.springframework=DEBUG
```

3.4.1 yml配置文件

> yml 是一种新型的类似于 xml 的文件类型,它去除了繁琐的标签,依赖于:来区分层级结构,可以方便的去描述信息,在 springboot 中 yml 文件和properties一样都可以用于配置 springboot,配置的选项一致,注意:和值之间需要有一个空格,注释是#需要在内容之前加

```
#相当于 properties 中的server.port=80
server:
  port: 80
#代表 spring.jpa.下面的属性 每个:之后的代表当前属性下的属性
spring:
  jpa:
    generate-ddl: false
    show-sql: true
    hibernate:
      ddl-auto: update
    database: mysql
    spring.jpa.properties.hibernate.dialect: org.hibernate.dialect.MySQL5Dialect
  datasource:
    url: jdbc:mysql://localhost:3306/cloud
    username: root
    password: qishimeiyoumima
    driver-class-name: com.mysql.jdbc.Driver
```

3.4.2 加载不同的 yml 文件

> 我们在 yml 中配置 springboot 需要的所有的设置,但是有时候 有一些配置我们需要区分生产环境还是开发环境, 比如我们在开发的时候连接的是一个数据库,在上线后连接的是另外的数据库,如果我们每次修改的话都要修改很多地方,那么怎么办呢,我们可以将需要修改的设置单独保存到一个文件中,根据环境不同给文件添加不同的后缀名,然后在通用的配置文件中指定要加载哪个后缀名的文件即可 例如

以下示例仅仅是为了展示加载不同文件用,对当前项目没有任何实际意义

3.4.2.1通用的 application.yml 文件

```
server:
    # context-path: /myboot
    session-timeout: 1800
    tomcat:
        max-threads: 1000
        min-spare-threads: 30
    port: 18080
    uri-encoding: utf-8
security:
  basic:
    enabled: false
spring:
```

```
  thymeleaf:
    mode: LEGACYHTML5
    cache: false
  jackson:
    time-zone: GMT+8
    date-format: yyyy-MM-dd HH:mm:ss
  profiles:
  #指定加载后缀为 pro 的文件,如果要加载其他的,只需要修改这里的名字即可
    active: pro
  http:
    multipart:
      max-file-size: 30Mb
      max-request-size: 30Mb
  devtools:
    restart:
      enabled: true
mybatis:
  configuration:
    map-underscore-to-camel-case: true
  mapper-locations: mybatis/**/*Mapper.xml
  typeAliasesPackage: com.bootdo.**.domain
#配置缓存和session存储方式，默认ehcache,可选redis
cacheType: ehcache
```

3.4.2.2 application-pro.yml

> 这个文件是以 pro 结束的文件,上面的设置加载的就是这个文件

```
bootdo:
  uploadPath: /var/uploaded_files/
logging:
  level:
    root: error
spring:
  datasource:
    type: com.alibaba.druid.pool.DruidDataSource
    driverClassName: com.mysql.jdbc.Driver
    #url: jdbc:mysql://127.0.0.1:3306/bootdo?useUnicode=true&characterEncoding=utf8
    url: jdbc:mysql://rm-uf68m0u6742ebeituo.mysql.rds.aliyuncs.com/bootdo?
useUnicode=true&characterEncoding=utf8
    username: root
    password: MYSQLmima001
    initialSize: 1
    minIdle: 3
    maxActive: 20
    # 配置获取连接等待超时的时间
    maxWait: 60000
    # 配置间隔多久才进行一次检测，检测需要关闭的空闲连接，单位是毫秒
    timeBetweenEvictionRunsMillis: 60000
```

```
    # 配置一个连接在池中最小生存的时间，单位是毫秒
    minEvictableIdleTimeMillis: 30000
    validationQuery: select 'x'
    testWhileIdle: true
    testOnBorrow: false
    testOnReturn: false
    # 打开PSCache，并且指定每个连接上PSCache的大小
    poolPreparedStatements: true
    maxPoolPreparedStatementPerConnectionSize: 20
    # 配置监控统计拦截的filters，去掉后监控界面sql无法统计，'wall'用于防火墙
    filters: stat,wall,slf4j
    # 通过connectProperties属性来打开mergeSql功能；慢SQL记录
    connectionProperties: druid.stat.mergeSql=true;druid.stat.slowSqlMillis=5000
    # 合并多个DruidDataSource的监控数据
    #useGlobalDataSourceStat: true
  redis:
      host: localhost
      port: 6379
      password:
      # 连接超时时间（毫秒）
      timeout: 10000
      pool:
        # 连接池中的最大空闲连接
        max-idle: 8
        # 连接池中的最小空闲连接
        min-idle: 10
        # 连接池最大连接数（使用负值表示没有限制）
        max-active: 100
        # 连接池最大阻塞等待时间（使用负值表示没有限制）
        max-wait: -1
```

3.4.2.3 application-dev.yml

> 这是另外一个文件,用于在开发环境下使用

```
bootdo:
  uploadPath: D:/var/uploaded_files/
logging:
  level:
    com.bootdo: debug
spring:
  datasource:
    type: com.alibaba.druid.pool.DruidDataSource
    driverClassName: com.mysql.jdbc.Driver
    url: jdbc:mysql://127.0.0.1:3306/bootdo?useUnicode=true&characterEncoding=utf8
    username: root
    password: root
    initialSize: 1
    minIdle: 3
```

```
    maxActive: 20
    # 配置获取连接等待超时的时间
    maxWait: 60000
    # 配置间隔多久才进行一次检测，检测需要关闭的空闲连接，单位是毫秒
    timeBetweenEvictionRunsMillis: 60000
    # 配置一个连接在池中最小生存的时间，单位是毫秒
    minEvictableIdleTimeMillis: 30000
    validationQuery: select 'x'
    testWhileIdle: true
    testOnBorrow: false
    testOnReturn: false
    # 打开PSCache，并且指定每个连接上PSCache的大小
    poolPreparedStatements: true
    maxPoolPreparedStatementPerConnectionSize: 20
    # 配置监控统计拦截的filters，去掉后监控界面sql无法统计，'wall'用于防火墙
    filters: stat,wall,slf4j
    # 通过connectProperties属性来打开mergeSql功能；慢SQL记录
    connectionProperties: druid.stat.mergeSql=true;druid.stat.slowSqlMillis=5000
    # 合并多个DruidDataSource的监控数据
    #useGlobalDataSourceStat: true
  redis:
      host: localhost
      port: 6379
      password:
      # 连接超时时间（毫秒）
      timeout: 10000
      pool:
        # 连接池中的最大空闲连接
        max-idle: 8
        # 连接池中的最小空闲连接
        min-idle: 10
        # 连接池最大连接数（使用负值表示没有限制）
        max-active: 100
        # 连接池最大阻塞等待时间（使用负值表示没有限制）
        max-wait: -1
```

> 所有的可以配置的属性请参见附录1

3.5 starter 启动器

springboot 提供了很多 starter，可以帮我们快速构建相应的开发环境,导入相关的依赖,并设置相关配置

Name	Description	Pom
spring-boot-	Core starter, including auto-configuration support, logging and YAML 核心启动器,包括自动配置,日志以及	Pom

`starter`	YAML 支持	
`spring-boot-starter-activemq`	Starter for JMS messaging using Apache ActiveMQ 对Apache ActiveMQ提供支持	Pom
`spring-boot-starter-amqp`	Starter for using Spring AMQP and Rabbit MQ 对Spring AMQP 和 Rabbit MQ支持	Pom
`spring-boot-starter-aop`	Starter for aspect-oriented programming with Spring AOP and AspectJ 对 spring AOP和AspectJ提供支持	Pom
`spring-boot-starter-artemis`	Starter for JMS messaging using Apache Artemis 对Apache Artemis提供支持	Pom
`spring-boot-starter-batch`	Starter for using Spring Batch	Pom
`spring-boot-starter-cache`	Starter for using Spring Framework's caching support	Pom
`spring-boot-starter-cloud-connectors`	Starter for using Spring Cloud Connectors which simplifies connecting to services in cloud platforms like Cloud Foundry and Heroku	Pom
`spring-boot-starter-data-cassandra`	Starter for using Cassandra distributed database and Spring Data Cassandra	Pom
`spring-boot-starter-data-cassandra-reactive`	Starter for using Cassandra distributed database and Spring Data Cassandra Reactive	Pom
`spring-boot-starter-data-couchbase`	Starter for using Couchbase document-oriented database and Spring Data Couchbase	Pom
`spring-boot-starter-data-couchbase-reactive`	Starter for using Couchbase document-oriented database and Spring Data Couchbase Reactive	Pom
`spring-boot-starter-data-elasticsearch`	Starter for using Elasticsearch search and analytics engine and Spring Data Elasticsearch	Pom
`spring-boot-starter-data-jpa`	Starter for using Spring Data JPA with Hibernate	Pom
`spring-boot-`		

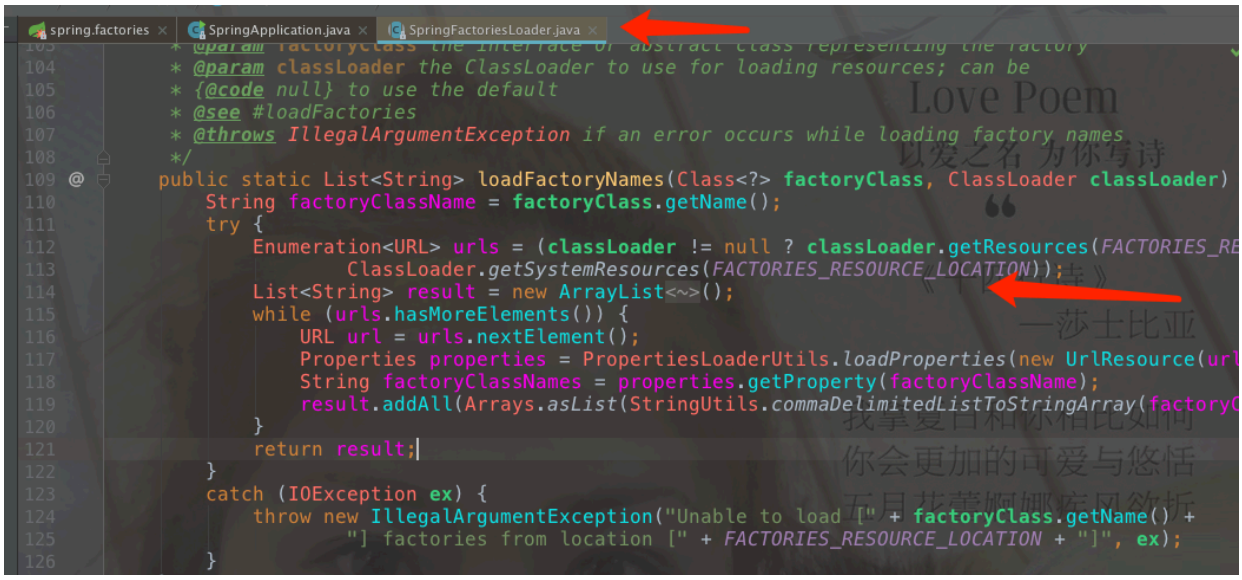
`starter-data-ldap`	Starter for using Spring Data LDAP	Pom
`spring-boot-starter-data-mongodb`	Starter for using MongoDB document-oriented database and Spring Data MongoDB	Pom
`spring-boot-starter-data-mongodb-reactive`	Starter for using MongoDB document-oriented database and Spring Data MongoDB Reactive	Pom
`spring-boot-starter-data-neo4j`	Starter for using Neo4j graph database and Spring Data Neo4j	Pom
`spring-boot-starter-data-redis`	Starter for using Redis key-value data store with Spring Data Redis and the Lettuce client	Pom
`spring-boot-starter-data-redis-reactive`	Starter for using Redis key-value data store with Spring Data Redis reactive and the Lettuce client	Pom
`spring-boot-starter-data-rest`	Starter for exposing Spring Data repositories over REST using Spring Data REST	Pom
`spring-boot-starter-data-solr`	Starter for using the Apache Solr search platform with Spring Data Solr	Pom
`spring-boot-starter-freemarker`	Starter for building MVC web applications using FreeMarker views	Pom
`spring-boot-starter-groovy-templates`	Starter for building MVC web applications using Groovy Templates views	Pom
`spring-boot-starter-hateoas`	Starter for building hypermedia-based RESTful web application with Spring MVC and Spring HATEOAS	Pom
`spring-boot-starter-integration`	Starter for using Spring Integration	Pom
`spring-boot-starter-jdbc`	Starter for using JDBC with the Tomcat JDBC connection pool	Pom
`spring-boot-starter-jersey`	Starter for building RESTful web applications using JAX-RS and Jersey. An alternative to `spring-boot-starter-web`	Pom
	Starter for using jOOQ to access SQL databases. An	

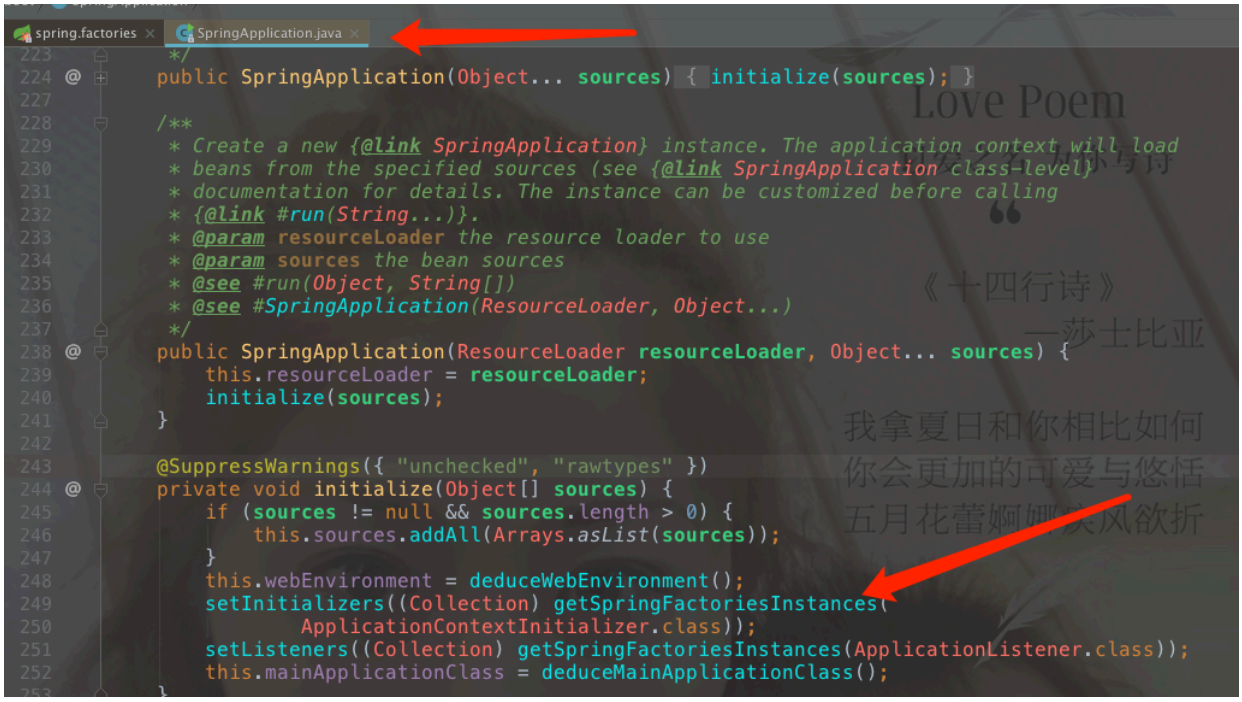
`spring-boot-starter-jooq`	alternative to `spring-boot-starter-data-jpa` or `spring-boot-starter-jdbc`	Pom
`spring-boot-starter-json`	Starter for reading and writing json	Pom
`spring-boot-starter-jta-atomikos`	Starter for JTA transactions using Atomikos	Pom
`spring-boot-starter-jta-bitronix`	Starter for JTA transactions using Bitronix	Pom
`spring-boot-starter-jta-narayana`	Spring Boot Narayana JTA Starter	Pom
`spring-boot-starter-mail`	Starter for using Java Mail and Spring Framework's email sending support	Pom
`spring-boot-starter-mustache`	Starter for building web applications using Mustache views	Pom
`spring-boot-starter-quartz`	Spring Boot Quartz Starter	Pom
`spring-boot-starter-security`	Starter for using Spring Security	Pom
`spring-boot-starter-test`	Starter for testing Spring Boot applications with libraries including JUnit, Hamcrest and Mockito	Pom
`spring-boot-starter-thymeleaf`	Starter for building MVC web applications using Thymeleaf views	Pom
`spring-boot-starter-validation`	Starter for using Java Bean Validation with Hibernate Validator	Pom
`spring-boot-starter-web`	Starter for building web, including RESTful, applications using Spring MVC. Uses Tomcat as the default embedded container	Pom
`spring-boot-starter-web-services`	Starter for using Spring Web Services	Pom
`spring-boot-`	Starter for building WebFlux applications using Spring	Pom

`starter-webflux`	Framework's Reactive Web support	
`spring-boot-starter-websocket`	Starter for building WebSocket applications using Spring Framework's WebSocket support	Pom

四. 自动装配的实现

Spring Boot在进行SpringApplication对象实例化时会加载spring-boot依赖包下面的META-INF/spring.factories文件，将该配置文件中的配置载入到Spring容器。spring.factories参见附录2





五 spring 入门案例之整合 mybatis

5.1 pom 依赖文件

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.kfit</groupId>
  <artifactId>spring-boot-mybatis</artifactId>
```

```
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>spring-boot-mybatis</name>
  <url>http://maven.apache.org</url>

  <!-- spring boot parent节点，引入这个之后，在下面和spring boot相关的就不需要引入版本了; -->
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>1.5.9.RELEASE</version>
    </parent>


  <dependencies>

        <!-- web支持: 1、web mvc; 2、restful; 3、jackjson支持; 4、aop ........ -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <!-- mysql 数据库驱动. -->
        <dependency>
                <groupId>mysql</groupId>
                <artifactId>mysql-connector-java</artifactId>
        </dependency>

        <!--
            spring-boot mybatis依赖：

            请不要使用1.0.0版本，因为还不支持拦截器插件，
            1.1.1 是博主写帖子时候的版本，大家使用最新版本即可
         -->

        <dependency>
            <groupId>org.mybatis.spring.boot</groupId>
            <artifactId>mybatis-spring-boot-starter</artifactId>
            <version>1.3.1</version>
        </dependency>



        <!--
        MyBatis提供了拦截器接口，我们可以实现自己的拦截器，
        将其作为一个plugin装入到SqlSessionFactory中。
        Github上有位开发者写了一个分页插件，我觉得使用起来还可以，挺方便的。
        Github项目地址：  https://github.com/pagehelper/Mybatis-PageHelper
```

```
        它同时提供了 springboot 的快速模式

      <dependency>
          <groupId>com.github.pagehelper</groupId>
          <artifactId>pagehelper</artifactId>
          <version>5.0.2</version>
      </dependency>
     -->


    <!-- https://mvnrepository.com/artifact/com.github.pagehelper/pagehelper-
spring-boot-starter
    pagerhelper 的 springboot 配置方式
-->
      <dependency>
          <groupId>com.github.pagehelper</groupId>
          <artifactId>pagehelper-spring-boot-starter</artifactId>
          <version>1.2.3</version>
      </dependency>


  </dependencies>
</project>
```

5.2 application.properties

```
spring.datasource.url = jdbc:mysql://localhost:3306/test
spring.datasource.username = root
spring.datasource.password = qishimeiyoumima
spring.datasource.driverClassName = com.mysql.jdbc.Driver
spring.datasource.max-active=20
spring.datasource.max-idle=8
spring.datasource.min-idle=8
spring.datasource.initial-size=10
spring.http.encoding.charset= UTF-8
```

5.3 pojo

```
public class Demo {
    private long id;
    private String name;
    public long getId() {
        return id;
    }
    public void setId(long id) {
        this.id = id;
```

```
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }

}
```

5.4 mapper

```
//mybatis在 springboot 项目中可以通过注解的方式使用 sql 语句
public interface DemoMappper {

    //#{name}:参数占位符
    @Select("select *from Demo where name=#{name}")
    public List<Demo> likeName(String name);


    @Select("select *from Demo where id = #{id}")
    public Demo getById(long id);

    @Select("select name from Demo where id = #{id}")
    public String getNameById(long id);


    /**
     * 保存数据.
     */
    @Insert("insert into Demo(name) values(#{name})")
    @Options(useGeneratedKeys=true,keyProperty="id",keyColumn="id")
    public void save(Demo demo);

}
```

5.5 service

```
@Service
public class DemoService {

    @Autowired
    private DemoMappper demoMappper;
```

```
    public List<Demo> likeName(String name){
        return demoMappper.likeName(name);
    }

    @Transactional//添加事务,在方法上面是给当前方法添加,在类中是给类中的所有方法添加
    public void save(Demo demo){
        demoMappper.save(demo);
    }

}
```

5.6 controller

```
@RestController//声明式一个 rest 风格的 controller
public class DemoController {

    @Autowired
    private DemoService demoService;

    @RequestMapping(value = "/likeName/{name}")
    public List<Demo> likeName(@PathVariable("name") String name){
        /*
         * 第一个参数：第几页;
         * 第二个参数：每页获取的条数.
         */
        PageHelper.startPage(1, 2);
        return demoService.likeName(name);
    }

    @RequestMapping("/save")
    public Demo save(){
        Demo demo = new Demo();
        demo.setName("张三");
        demoService.save(demo);
        return demo;
    }

}
```

5.7测试

```
访问http://localhost:8080/save
```

访问 http://localhost:8080/likeName/张三

六 配置全局异常捕获

要捕获全局异常只需要以下几步即可
1. 创建一个类,在类上面添加@ControllerAdvice注解
2.编写任意一个方法,参数是HttpServletRequest和Exception,在方法上面添加@ExceptionHandler注解,方法返回值如果是字符串,则还需啊添加@ResponseBody,如果返回的是页面,则返回ModelAndView 对象
3.按照自己的业务逻辑决定返回什么

```
/**
 * Created by jackiechan on 2017/12/31/下午11:30
 */
@ControllerAdvice
public class GloablExceptionHandler {
    @ExceptionHandler
    @ResponseBody
    public String exceptionHandler(HttpServletRequest request, Exception e){
        return "错误";
    }
}
```

七 SpringBoot 热部署

7.1 添加插件

```
<build>
 <plugins>
 <plugin>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-maven-plugin</artifactId>
 </plugin>
 </plugins>
</build>
```

7.2 添加相关依赖

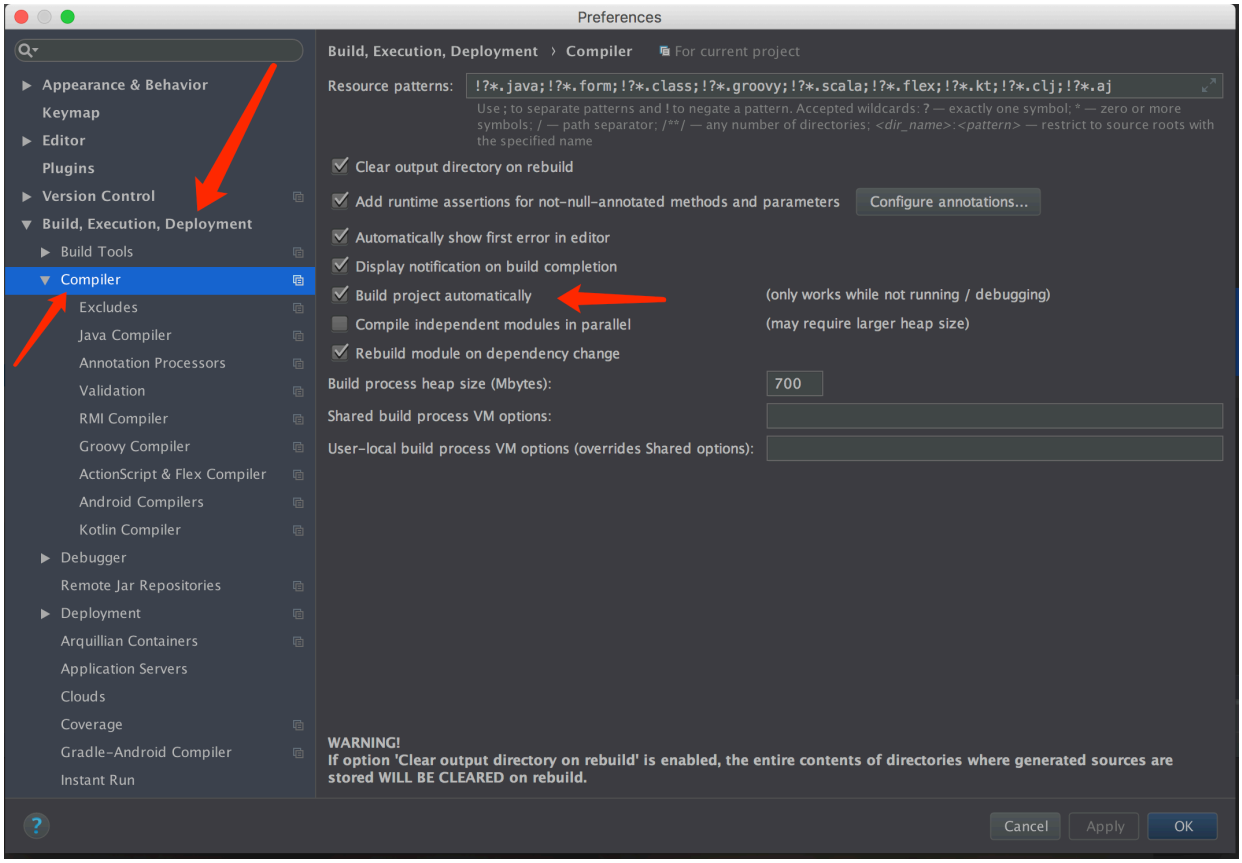
```
<dependency>
 <groupId>org.springframework.boot</groupId>
 <artifactId>spring-boot-devtools</artifactId>
 <optional>true</optional> <!-- 这个需要为 true 热部署才有效 -->
</dependency>
```

7.3 启动项目

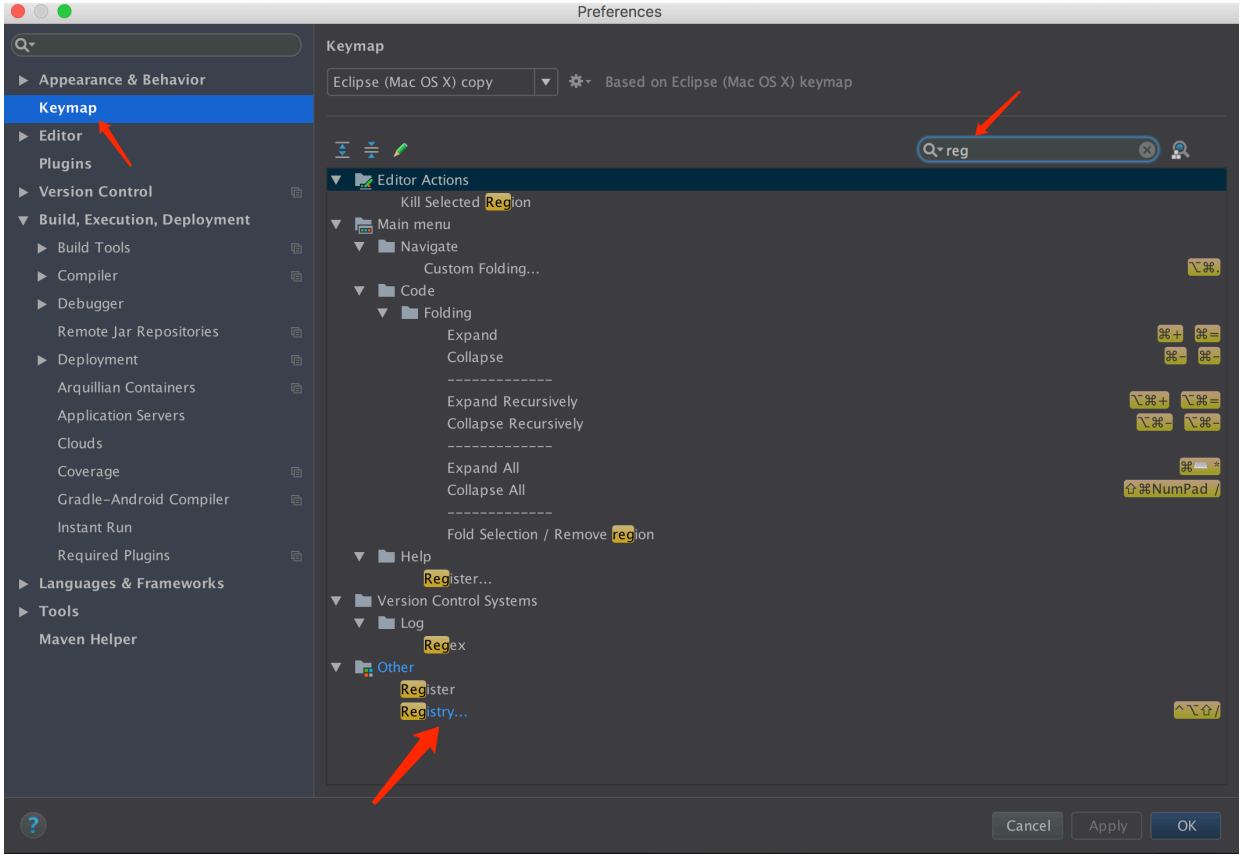
```
通过使用 maven 的spring-boot:run指令启动项目,项目后续的 class 发生变化既可热部署进来
```

7.4 Idea 开启自动编译

7.4.1 开启自动编译

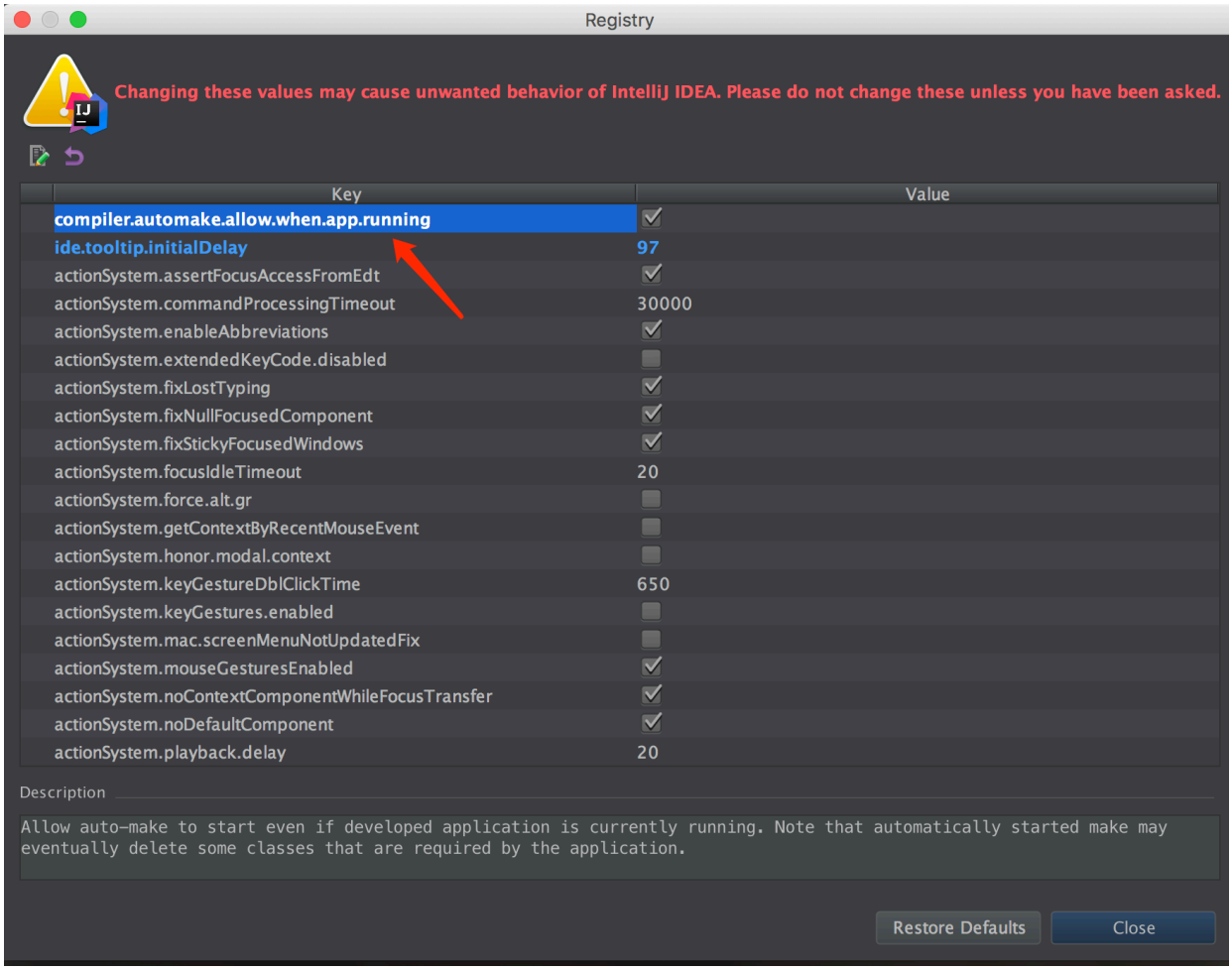


7.4.2 设置快键键



7.4.3 开启运行时自动编译

通过使用上面配置的快捷键唤出下面窗口,找到 c 开头的下面选项,选中即可



附录

附录1 springboot 全局配置属性

```
# ===================================================================
# 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)
spring.application.index= # Application index.
spring.application.name= # Application name.

# ADMIN (SpringApplicationAdminJmxAutoConfiguration)
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)
spring.cache.cache-names= # Comma-separated list of cache names to create if
supported by the underlying cache manager.
spring.cache.caffeine.spec= # The spec to use to create caches. Check CaffeineSpec
for more details on the spec format.
spring.cache.couchbase.expiration=0 # Entry expiration in milliseconds. By default
the entries never expire.
spring.cache.ehcache.config= # The location of the configuration file to use to
initialize EhCache.
spring.cache.guava.spec= # The spec to use to create caches. Check
CacheBuilderSpec for more details on the spec format.
spring.cache.infinispan.config= # The location of the configuration file to use to
initialize Infinispan.
spring.cache.jcache.config= # The location of the configuration file to use to
initialize the cache manager.
spring.cache.jcache.provider= # Fully qualified name of the CachingProvider
implementation to use to retrieve the JSR-107 compliant cache manager. Only needed
if more than one JSR-107 implementation is available on the classpath.
spring.cache.type= # Cache type, auto-detected according to the environment by
default.

# SPRING CONFIG - using environment property only (ConfigFileApplicationListener)
spring.config.location= # Config file locations.
spring.config.name=application # Config file name.

# HAZELCAST (HazelcastProperties)
spring.hazelcast.config= # The location of the configuration file to use to
initialize Hazelcast.

# PROJECT INFORMATION (ProjectInfoProperties)
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)
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)
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)
spring.mandatory-file-encoding= # Expected character encoding the application must
use.

# INTERNATIONALIZATION (MessageSourceAutoConfiguration)
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)
spring.pid.fail-on-write-error= # Fail if ApplicationPidFileWriter is used but it
cannot write the PID file.
spring.pid.file= # Location of the PID file to write (if ApplicationPidFileWriter
is used).

# PROFILES
spring.profiles.active= # Comma-separated list (or list if using YAML) of active
profiles.
```

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

# SENDGRID (SendGridAutoConfiguration)
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)
server.address= # Network address to which the server should bind to.
server.compression.enabled=false # If response compression is enabled.
server.compression.excluded-user-agents= # List of user-agents to exclude from
compression.
server.compression.mime-types= # Comma-separated list of MIME types that should be
compressed. For instance `text/html,text/css,application/json`
server.compression.min-response-size= # Minimum response size that is required for
compression to be performed. For instance 2048
server.connection-timeout= # Time in milliseconds that connectors will wait for
another HTTP request before closing the connection. When not set, the connector's
container-specific default will be used. Use a value of -1 to indicate no (i.e.
infinite) timeout.
server.context-parameters.*= # Servlet context init parameters. For instance
`server.context-parameters.a=alpha`
server.context-path= # Context path of the application.
server.display-name=application # Display name of the application.
server.max-http-header-size=0 # Maximum size in bytes of the HTTP message header.
server.error.include-stacktrace=never # When to include a "stacktrace" attribute.
server.error.path=/error # Path of the error controller.
server.error.whitelabel.enabled=true # Enable the default error page displayed in
browsers in case of a server error.
server.jetty.acceptors= # Number of acceptor threads to use.
server.jetty.max-http-post-size=0 # Maximum size in bytes of the HTTP post or put
content.
server.jetty.selectors= # Number of selector threads to use.
server.jsp-servlet.class-name=org.apache.jasper.servlet.JspServlet # The class
name of the JSP servlet.
server.jsp-servlet.init-parameters.*= # Init parameters used to configure the JSP
servlet
server.jsp-servlet.registered=true # Whether or not the JSP servlet is registered
server.port=8080 # Server HTTP port.
server.server-header= # Value to use for the Server response header (no header is
sent if empty)
```

```
server.servlet-path=/ # Path of the main dispatcher servlet.
server.use-forward-headers= # If X-Forwarded-* headers should be applied to the
HttpRequest.
server.session.cookie.comment= # Comment for the session cookie.
server.session.cookie.domain= # Domain for the session cookie.
server.session.cookie.http-only= # "HttpOnly" flag for the session cookie.
server.session.cookie.max-age= # Maximum age of the session cookie in seconds.
server.session.cookie.name= # Session cookie name.
server.session.cookie.path= # Path of the session cookie.
server.session.cookie.secure= # "Secure" flag for the session cookie.
server.session.persistent=false # Persist session data between restarts.
server.session.store-dir= # Directory used to store session data.
server.session.timeout= # Session timeout in seconds.
server.session.tracking-modes= # Session tracking modes (one or more of the
following: "cookie", "url", "ssl").
server.ssl.ciphers= # Supported SSL ciphers.
server.ssl.client-auth= # Whether client authentication is wanted ("want") or
needed ("need"). Requires a trust store.
server.ssl.enabled= # Enable SSL support.
server.ssl.enabled-protocols= # Enabled SSL protocols.
server.ssl.key-alias= # Alias that identifies the key in the key store.
server.ssl.key-password= # Password used to access the key in the key store.
server.ssl.key-store= # Path to the key store that holds the SSL certificate
(typically a jks file).
server.ssl.key-store-password= # Password used to access the key store.
server.ssl.key-store-provider= # Provider for the key store.
server.ssl.key-store-type= # Type of the key store.
server.ssl.protocol=TLS # SSL protocol to use.
server.ssl.trust-store= # Trust store that holds SSL certificates.
server.ssl.trust-store-password= # Password used to access the trust store.
server.ssl.trust-store-provider= # Provider for the trust store.
server.ssl.trust-store-type= # Type of the trust store.
server.tomcat.accept-count= # Maximum queue length for incoming connection
requests when all possible request processing threads are in use.
server.tomcat.accesslog.buffered=true # Buffer output such that it is only flushed
periodically.
server.tomcat.accesslog.directory=logs # Directory in which log files are created.
Can be relative to the tomcat base dir or absolute.
server.tomcat.accesslog.enabled=false # Enable access log.
server.tomcat.accesslog.pattern=common # Format pattern for access logs.
server.tomcat.accesslog.prefix=access_log # Log file name prefix.
server.tomcat.accesslog.rename-on-rotate=false # Defer inclusion of the date stamp
in the file name until rotate time.
server.tomcat.accesslog.request-attributes-enabled=false # Set request attributes
for IP address, Hostname, protocol and port used for the request.
server.tomcat.accesslog.rotate=true # Enable access log rotation.
server.tomcat.accesslog.suffix=.log # Log file name suffix.
server.tomcat.additional-tld-skip-patterns= # Comma-separated list of additional
patterns that match jars to ignore for TLD scanning.
```

```
server.tomcat.background-processor-delay=30 # Delay in seconds between the
invocation of backgroundProcess methods.
server.tomcat.basedir= # Tomcat base directory. If not specified a temporary
directory will be used.
server.tomcat.internal-proxies=10\\.\\d{1,3}\\.\\d{1,3}\\.\\d{1,3}|\\
        192\\.168\\.\\d{1,3}\\.\\d{1,3}|\\
        169\\.254\\.\\d{1,3}\\.\\d{1,3}|\\
        127\\.\\d{1,3}\\.\\d{1,3}\\.\\d{1,3}|\\
        172\\.1[6-9]{1}\\.\\d{1,3}\\.\\d{1,3}|\\
        172\\.2[0-9]{1}\\.\\d{1,3}\\.\\d{1,3}|\\
        172\\.3[0-1]{1}\\.\\d{1,3}\\.\\d{1,3} # regular expression matching
trusted IP addresses.
server.tomcat.max-connections= # Maximum number of connections that the server
will accept and process at any given time.
server.tomcat.max-http-post-size=0 # Maximum size in bytes of the HTTP post
content.
server.tomcat.max-threads=0 # Maximum amount of worker threads.
server.tomcat.min-spare-threads=0 # Minimum amount of worker threads.
server.tomcat.port-header=X-Forwarded-Port # Name of the HTTP header used to
override the original port value.
server.tomcat.protocol-header= # Header that holds the incoming protocol, usually
named "X-Forwarded-Proto".
server.tomcat.protocol-header-https-value=https # Value of the protocol header
that indicates that the incoming request uses SSL.
server.tomcat.redirect-context-root= # Whether requests to the context root should
be redirected by appending a / to the path.
server.tomcat.remote-ip-header= # Name of the http header from which the remote ip
is extracted. For instance `X-FORWARDED-FOR`
server.tomcat.uri-encoding=UTF-8 # Character encoding to use to decode the URI.
server.undertow.accesslog.dir= # Undertow access log directory.
server.undertow.accesslog.enabled=false # Enable access log.
server.undertow.accesslog.pattern=common # Format pattern for access logs.
server.undertow.accesslog.prefix=access_log. # Log file name prefix.
server.undertow.accesslog.rotate=true # Enable access log rotation.
server.undertow.accesslog.suffix=log # Log file name suffix.
server.undertow.buffer-size= # Size of each buffer in bytes.
server.undertow.buffers-per-region= # Number of buffer per region.
server.undertow.direct-buffers= # Allocate buffers outside the Java heap.
server.undertow.io-threads= # Number of I/O threads to create for the worker.
server.undertow.max-http-post-size=0 # Maximum size in bytes of the HTTP post
content.
server.undertow.worker-threads= # Number of worker threads.

# FREEMARKER (FreeMarkerAutoConfiguration)
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)
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)
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)
spring.http.encoding.charset=UTF-8 # Charset of HTTP requests and responses. Added
to the "Content-Type" header if not set explicitly.
spring.http.encoding.enabled=true # Enable http encoding support.
spring.http.encoding.force= # Force the encoding to the configured charset on HTTP
requests and responses.
spring.http.encoding.force-request= # Force the encoding to the configured charset
on HTTP requests. Defaults to true when "force" has not been specified.
spring.http.encoding.force-response= # Force the encoding to the configured
charset on HTTP responses.
spring.http.encoding.mapping= # Locale to Encoding mapping.

# MULTIPART (MultipartProperties)
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)
spring.jackson.date-format= # Date format string or a fully-qualified date format
class name. For instance `yyyy-MM-dd HH:mm:ss`.
spring.jackson.default-property-inclusion= # Controls the inclusion of properties
during serialization.
spring.jackson.deserialization.*= # Jackson on/off features that affect the way
Java objects are deserialized.
spring.jackson.generator.*= # Jackson on/off features for generators.
spring.jackson.joda-date-time-format= # Joda date time format string. If not
configured, "date-format" will be used as a fallback if it is configured with a
format string.
spring.jackson.locale= # Locale used for formatting.
spring.jackson.mapper.*= # Jackson general purpose on/off features.
spring.jackson.parser.*= # Jackson on/off features for parsers.
spring.jackson.property-naming-strategy= # One of the constants on Jackson's
PropertyNamingStrategy. Can also be a fully-qualified class name of a
PropertyNamingStrategy subclass.
spring.jackson.serialization.*= # Jackson on/off features that affect the way Java
objects are serialized.
spring.jackson.time-zone= # Time zone used when formatting dates. For instance
`America/Los_Angeles`

# JERSEY (JerseyProperties)
spring.jersey.application-path= # Path that serves as the base URI for the
application. Overrides the value of "@ApplicationPath" if specified.
spring.jersey.filter.order=0 # Jersey filter chain order.
spring.jersey.init.*= # Init parameters to pass to Jersey via the servlet or
filter.
spring.jersey.servlet.load-on-startup=-1 # Load on startup priority of the Jersey
servlet.
spring.jersey.type=servlet # Jersey integration type.

# SPRING LDAP (LdapProperties)
spring.ldap.urls= # LDAP URLs of the server.
spring.ldap.base= # Base suffix from which all operations should originate.
spring.ldap.username= # Login user of the server.
spring.ldap.password= # Login password of the server.
spring.ldap.base-environment.*= # LDAP specification settings.

# EMBEDDED LDAP (EmbeddedLdapProperties)
spring.ldap.embedded.base-dn= # The base DN
spring.ldap.embedded.credential.username= # Embedded LDAP username.
spring.ldap.embedded.credential.password= # Embedded LDAP password.
spring.ldap.embedded.ldif=classpath:schema.ldif # Schema (LDIF) script resource
reference.
spring.ldap.embedded.port= # Embedded LDAP port.
```

```
spring.ldap.embedded.validation.enabled=true # Enable LDAP schema validation.
spring.ldap.embedded.validation.schema= # Path to the custom schema.

# SPRING MOBILE DEVICE VIEWS (DeviceDelegatingViewResolverAutoConfiguration)
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)
spring.mobile.sitepreference.enabled=true # Enable SitePreferenceHandler.

# MUSTACHE TEMPLATES (MustacheAutoConfiguration)
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)
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)
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)
spring.session.hazelcast.flush-mode=on-save # Sessions flush mode.
spring.session.hazelcast.map-name=spring:session:sessions # Name of the map used
to store sessions.
spring.session.jdbc.initializer.enabled= # Create the required session tables on
startup if necessary. Enabled automatically if the default table name is set or a
custom schema is configured.
spring.session.jdbc.schema=classpath:org/springframework/session/jdbc/schema-
@@platform@@.sql # Path to the SQL file to use to initialize the database schema.
spring.session.jdbc.table-name=SPRING_SESSION # Name of database table used to
store sessions.
spring.session.mongo.collection-name=sessions # Collection name used to store
sessions.
spring.session.redis.flush-mode=on-save # Sessions flush mode.
spring.session.redis.namespace= # Namespace for keys used to store sessions.
spring.session.store-type= # Session store type.

# SPRING SOCIAL (SocialWebAutoConfiguration)
spring.social.auto-connection-views=false # Enable the connection status view for
supported providers.

# SPRING SOCIAL FACEBOOK (FacebookAutoConfiguration)
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)
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)
spring.social.twitter.app-id= # your application's Twitter App ID
spring.social.twitter.app-secret= # your application's Twitter App Secret

# THYMELEAF (ThymeleafAutoConfiguration)
spring.thymeleaf.cache=true # Enable template caching.
spring.thymeleaf.check-template=true # Check that the template exists before
rendering it.
spring.thymeleaf.check-template-location=true # Check that the templates location
exists.
spring.thymeleaf.content-type=text/html # Content-Type value.
spring.thymeleaf.enabled=true # Enable MVC Thymeleaf view resolution.
```

```
spring.thymeleaf.encoding=UTF-8 # Template encoding.
spring.thymeleaf.excluded-view-names= # Comma-separated list of view names that
should be excluded from resolution.
spring.thymeleaf.mode=HTML5 # Template mode to be applied to templates. See also
StandardTemplateModeHandlers.
spring.thymeleaf.prefix=classpath:/templates/ # Prefix that gets prepended to view
names when building a URL.
spring.thymeleaf.suffix=.html # Suffix that gets appended to view names when
building a URL.
spring.thymeleaf.template-resolver-order= # Order of the template resolver in the
chain.
spring.thymeleaf.view-names= # Comma-separated list of view names that can be
resolved.

# SPRING WEB SERVICES (WebServicesProperties)
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)
security.basic.authorize-mode=role # Security authorize mode to apply.
security.basic.enabled=true # Enable basic authentication.
security.basic.path=/** # Comma-separated list of paths to secure.
security.basic.realm=Spring # HTTP basic realm name.
security.enable-csrf=false # Enable Cross Site Request Forgery support.
security.filter-order=0 # Security filter chain order.
security.filter-dispatcher-types=ASYNC, FORWARD, INCLUDE, REQUEST # Security
filter chain dispatcher types.
security.headers.cache=true # Enable cache control HTTP headers.
security.headers.content-security-policy= # Value for content security policy
header.
security.headers.content-security-policy-mode=default # Content security policy
mode.
security.headers.content-type=true # Enable "X-Content-Type-Options" header.
security.headers.frame=true # Enable "X-Frame-Options" header.
security.headers.hsts=all # HTTP Strict Transport Security (HSTS) mode (none,
domain, all).
security.headers.xss=true # Enable cross site scripting (XSS) protection.
security.ignored= # Comma-separated list of paths to exclude from the default
secured paths.
security.require-ssl=false # Enable secure channel for all requests.
```

```
security.sessions=stateless # Session creation policy (always, never, if_required,
stateless).
security.user.name=user # Default user name.
security.user.password= # Password for the default user name. A random password is
logged on startup by default.
security.user.role=USER # Granted roles for the default user name.

# SECURITY OAUTH2 CLIENT (OAuth2ClientProperties)
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)
security.oauth2.resource.filter-order= # The order of the filter chain used to
authenticate tokens.
security.oauth2.resource.id= # Identifier of the resource.
security.oauth2.resource.jwt.key-uri= # The URI of the JWT token. Can be set if
the value is not available and the key is public.
security.oauth2.resource.jwt.key-value= # The verification key of the JWT token.
Can either be a symmetric secret or PEM-encoded RSA public key.
security.oauth2.resource.prefer-token-info=true # Use the token info, can be set
to false to use the user info.
security.oauth2.resource.service-id=resource #
security.oauth2.resource.token-info-uri= # URI of the token decoding endpoint.
security.oauth2.resource.token-type= # The token type to send when using the
userInfoUri.
security.oauth2.resource.user-info-uri= # URI of the user endpoint.

# SECURITY OAUTH2 SSO (OAuth2SsoProperties)
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)
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)
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)
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)
spring.dao.exceptiontranslation.enabled=true # Enable the PersistenceExceptionTranslationPostProcessor.

# CASSANDRA (CassandraProperties)
spring.data.cassandra.cluster-name= # Name of the Cassandra cluster.
spring.data.cassandra.compression=none # Compression supported by the Cassandra binary protocol.
spring.data.cassandra.connect-timeout-millis= # Socket option: connection time out.
spring.data.cassandra.consistency-level= # Queries consistency level.
spring.data.cassandra.contact-points=localhost # Comma-separated list of cluster node addresses.
spring.data.cassandra.fetch-size= # Queries default fetch size.
spring.data.cassandra.keyspace-name= # Keyspace name to use.
spring.data.cassandra.load-balancing-policy= # Class name of the load balancing policy.
spring.data.cassandra.port= # Port of the Cassandra server.
spring.data.cassandra.password= # Login password of the server.
spring.data.cassandra.read-timeout-millis= # Socket option: read time out.
spring.data.cassandra.reconnection-policy= # Reconnection policy class.
spring.data.cassandra.retry-policy= # Class name of the retry policy.
spring.data.cassandra.serial-consistency-level= # Queries serial consistency level.
spring.data.cassandra.schema-action=none # Schema action to take at startup.
spring.data.cassandra.ssl=false # Enable SSL support.
spring.data.cassandra.username= # Login user of the server.

# DATA COUCHBASE (CouchbaseDataProperties)
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)
spring.data.elasticsearch.cluster-name=elasticsearch # Elasticsearch cluster name.
spring.data.elasticsearch.cluster-nodes= # Comma-separated list of cluster node
addresses. If not specified, starts a client node.
spring.data.elasticsearch.properties.*= # Additional properties used to configure
the client.
spring.data.elasticsearch.repositories.enabled=true # Enable Elasticsearch
repositories.

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

# MONGODB (MongoProperties)
spring.data.mongodb.authentication-database= # Authentication database name.
spring.data.mongodb.database=test # Database name.
spring.data.mongodb.field-naming-strategy= # Fully qualified name of the
FieldNamingStrategy to use.
spring.data.mongodb.grid-fs-database= # GridFS database name.
spring.data.mongodb.host=localhost # Mongo server host. Cannot be set with uri.
spring.data.mongodb.password= # Login password of the mongo server. Cannot be set
with uri.
spring.data.mongodb.port=27017 # Mongo server port. Cannot be set with uri.
spring.data.mongodb.repositories.enabled=true # Enable Mongo repositories.
spring.data.mongodb.uri=mongodb://localhost/test # Mongo database URI. Cannot be
set with host, port and credentials.
spring.data.mongodb.username= # Login user of the mongo server. Cannot be set with
uri.

# DATA REDIS
spring.data.redis.repositories.enabled=true # Enable Redis repositories.

# NEO4J (Neo4jProperties)
spring.data.neo4j.compiler= # Compiler to use.
spring.data.neo4j.embedded.enabled=true # Enable embedded mode if the embedded
driver is available.
spring.data.neo4j.open-in-view=false # Register OpenSessionInViewInterceptor.
Binds a Neo4j Session to the thread for the entire processing of the request.
spring.data.neo4j.password= # Login password of the server.
spring.data.neo4j.repositories.enabled=true # Enable Neo4j repositories.
spring.data.neo4j.uri= # URI used by the driver. Auto-detected by default.
spring.data.neo4j.username= # Login user of the server.

# DATA REST (RepositoryRestProperties)
spring.data.rest.base-path= # Base path to be used by Spring Data REST to expose
repository resources.
spring.data.rest.default-page-size= # Default size of pages.
spring.data.rest.detection-strategy=default # Strategy to use to determine which
repositories get exposed.
```

```
spring.data.rest.enable-enum-translation= # Enable enum value translation via the
Spring Data REST default resource bundle.
spring.data.rest.limit-param-name= # Name of the URL query string parameter that
indicates how many results to return at once.
spring.data.rest.max-page-size= # Maximum size of pages.
spring.data.rest.page-param-name= # Name of the URL query string parameter that
indicates what page to return.
spring.data.rest.return-body-on-create= # Return a response body after creating an
entity.
spring.data.rest.return-body-on-update= # Return a response body after updating an
entity.
spring.data.rest.sort-param-name= # Name of the URL query string parameter that
indicates what direction to sort results.

# SOLR (SolrProperties)
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 & DataSourceProperties)
spring.datasource.continue-on-error=false # Do not stop if an error occurs while
initializing the database.
spring.datasource.data= # Data (DML) script resource references.
spring.datasource.data-username= # User of the database to execute DML scripts (if
different).
spring.datasource.data-password= # Password of the database to execute DML scripts
(if different).
spring.datasource.dbcp2.*= # Commons DBCP2 specific settings
spring.datasource.driver-class-name= # Fully qualified name of the JDBC driver.
Auto-detected based on the URL by default.
spring.datasource.generate-unique-name=false # Generate a random datasource name.
spring.datasource.hikari.*= # Hikari specific settings
spring.datasource.initialize=true # Populate the database using 'data.sql'.
spring.datasource.jmx-enabled=false # Enable JMX support (if provided by the
underlying pool).
spring.datasource.jndi-name= # JNDI location of the datasource. Class, url,
username & password are ignored when set.
spring.datasource.name=testdb # Name of the datasource.
spring.datasource.password= # Login password of the database.
spring.datasource.platform=all # Platform to use in the schema resource (schema-
${platform}.sql).
spring.datasource.schema= # Schema (DDL) script resource references.
spring.datasource.schema-username= # User of the database to execute DDL scripts
(if different).
spring.datasource.schema-password= # Password of the database to execute DDL
scripts (if different).
spring.datasource.separator=; # Statement separator in SQL initialization scripts.
spring.datasource.sql-script-encoding= # SQL scripts encoding.
```

```
spring.datasource.tomcat.*= # Tomcat datasource specific settings
spring.datasource.type= # Fully qualified name of the connection pool
implementation to use. By default, it is auto-detected from the classpath.
spring.datasource.url= # JDBC url of the database.
spring.datasource.username=

# JEST (Elasticsearch HTTP client) (JestProperties)
spring.elasticsearch.jest.connection-timeout=3000 # Connection timeout in
milliseconds.
spring.elasticsearch.jest.multi-threaded=true # Enable connection requests from
multiple execution threads.
spring.elasticsearch.jest.password= # Login password.
spring.elasticsearch.jest.proxy.host= # Proxy host the HTTP client should use.
spring.elasticsearch.jest.proxy.port= # Proxy port the HTTP client should use.
spring.elasticsearch.jest.read-timeout=3000 # Read timeout in milliseconds.
spring.elasticsearch.jest.uris=http://localhost:9200 # Comma-separated list of the
Elasticsearch instances to use.
spring.elasticsearch.jest.username= # Login user.

# H2 Web Console (H2ConsoleProperties)
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)
spring.jooq.sql-dialect= # SQLDialect JOOQ used when communicating with the
configured datasource. For instance `POSTGRES`

# JPA (JpaBaseConfiguration, HibernateJpaAutoConfiguration)
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)
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)
spring.jta.atomikos.connectionfactory.borrow-connection-timeout=30 # Timeout, in
seconds, for borrowing connections from the pool.
spring.jta.atomikos.connectionfactory.ignore-session-transacted-flag=true #
Whether or not to ignore the transacted flag when creating session.
spring.jta.atomikos.connectionfactory.local-transaction-mode=false # Whether or
not local transactions are desired.
spring.jta.atomikos.connectionfactory.maintenance-interval=60 # The time, in
seconds, between runs of the pool's maintenance thread.
spring.jta.atomikos.connectionfactory.max-idle-time=60 # The time, in seconds,
after which connections are cleaned up from the pool.
spring.jta.atomikos.connectionfactory.max-lifetime=0 # The time, in seconds, that
a connection can be pooled for before being destroyed. 0 denotes no limit.
spring.jta.atomikos.connectionfactory.max-pool-size=1 # The maximum size of the
pool.
spring.jta.atomikos.connectionfactory.min-pool-size=1 # The minimum size of the
pool.
spring.jta.atomikos.connectionfactory.reap-timeout=0 # The reap timeout, in
seconds, for borrowed connections. 0 denotes no limit.
spring.jta.atomikos.connectionfactory.unique-resource-name=jmsConnectionFactory #
The unique name used to identify the resource during recovery.
spring.jta.atomikos.datasource.borrow-connection-timeout=30 # Timeout, in seconds,
for borrowing connections from the pool.
spring.jta.atomikos.datasource.default-isolation-level= # Default isolation level
of connections provided by the pool.
spring.jta.atomikos.datasource.login-timeout= # Timeout, in seconds, for
establishing a database connection.
spring.jta.atomikos.datasource.maintenance-interval=60 # The time, in seconds,
between runs of the pool's maintenance thread.
spring.jta.atomikos.datasource.max-idle-time=60 # The time, in seconds, after
which connections are cleaned up from the pool.
spring.jta.atomikos.datasource.max-lifetime=0 # The time, in seconds, that a
connection can be pooled for before being destroyed. 0 denotes no limit.
spring.jta.atomikos.datasource.max-pool-size=1 # The maximum size of the pool.
spring.jta.atomikos.datasource.min-pool-size=1 # The minimum size of the pool.
spring.jta.atomikos.datasource.reap-timeout=0 # The reap timeout, in seconds, for
borrowed connections. 0 denotes no limit.
spring.jta.atomikos.datasource.test-query= # SQL query or statement used to
validate a connection before returning it.
```

```
spring.jta.atomikos.datasource.unique-resource-name=dataSource # The unique name
used to identify the resource during recovery.
spring.jta.atomikos.properties.checkpoint-interval=500 # Interval between
checkpoints.
spring.jta.atomikos.properties.console-file-count=1 # Number of debug logs files
that can be created.
spring.jta.atomikos.properties.console-file-limit=-1 # How many bytes can be
stored at most in debug logs files.
spring.jta.atomikos.properties.console-file-name=tm.out # Debug logs file name.
spring.jta.atomikos.properties.console-log-level=warn # Console log level.
spring.jta.atomikos.properties.default-jta-timeout=10000 # Default timeout for JTA
transactions.
spring.jta.atomikos.properties.enable-logging=true # Enable disk logging.
spring.jta.atomikos.properties.force-shutdown-on-vm-exit=false # Specify if a VM
shutdown should trigger forced shutdown of the transaction core.
spring.jta.atomikos.properties.log-base-dir= # Directory in which the log files
should be stored.
spring.jta.atomikos.properties.log-base-name=tmlog # Transactions log file base
name.
spring.jta.atomikos.properties.max-actives=50 # Maximum number of active
transactions.
spring.jta.atomikos.properties.max-timeout=300000 # Maximum timeout (in
milliseconds) that can be allowed for transactions.
spring.jta.atomikos.properties.output-dir= # Directory in which to store the debug
log files.
spring.jta.atomikos.properties.serial-jta-transactions=true # Specify if sub-
transactions should be joined when possible.
spring.jta.atomikos.properties.service= # Transaction manager implementation that
should be started.
spring.jta.atomikos.properties.threaded-two-phase-commit=true # Use different (and
concurrent) threads for two-phase commit on the participating resources.
spring.jta.atomikos.properties.transaction-manager-unique-name= # Transaction
manager's unique name.

# BITRONIX
spring.jta.bitronix.connectionfactory.acquire-increment=1 # Number of connections
to create when growing the pool.
spring.jta.bitronix.connectionfactory.acquisition-interval=1 # Time, in seconds,
to wait before trying to acquire a connection again after an invalid connection
was acquired.
spring.jta.bitronix.connectionfactory.acquisition-timeout=30 # Timeout, in
seconds, for acquiring connections from the pool.
spring.jta.bitronix.connectionfactory.allow-local-transactions=true # Whether or
not the transaction manager should allow mixing XA and non-XA transactions.
spring.jta.bitronix.connectionfactory.apply-transaction-timeout=false # Whether or
not the transaction timeout should be set on the XAResource when it is enlisted.
spring.jta.bitronix.connectionfactory.automatic-enlisting-enabled=true # Whether
or not resources should be enlisted and delisted automatically.
```

```
spring.jta.bitronix.connectionfactory.cache-producers-consumers=true # Whether or not produces and consumers should be cached.
spring.jta.bitronix.connectionfactory.defer-connection-release=true # Whether or not the provider can run many transactions on the same connection and supports transaction interleaving.
spring.jta.bitronix.connectionfactory.ignore-recovery-failures=false # Whether or not recovery failures should be ignored.
spring.jta.bitronix.connectionfactory.max-idle-time=60 # The time, in seconds, after which connections are cleaned up from the pool.
spring.jta.bitronix.connectionfactory.max-pool-size=10 # The maximum size of the pool. 0 denotes no limit.
spring.jta.bitronix.connectionfactory.min-pool-size=0 # The minimum size of the pool.
spring.jta.bitronix.connectionfactory.password= # The password to use to connect to the JMS provider.
spring.jta.bitronix.connectionfactory.share-transaction-connections=false # Whether or not connections in the ACCESSIBLE state can be shared within the context of a transaction.
spring.jta.bitronix.connectionfactory.test-connections=true # Whether or not connections should be tested when acquired from the pool.
spring.jta.bitronix.connectionfactory.two-pc-ordering-position=1 # The position that this resource should take during two-phase commit (always first is Integer.MIN_VALUE, always last is Integer.MAX_VALUE).
spring.jta.bitronix.connectionfactory.unique-name=jmsConnectionFactory # The unique name used to identify the resource during recovery.
spring.jta.bitronix.connectionfactory.use-tm-join=true Whether or not TMJOIN should be used when starting XAResources.
spring.jta.bitronix.connectionfactory.user= # The user to use to connect to the JMS provider.
spring.jta.bitronix.datasource.acquire-increment=1 # Number of connections to create when growing the pool.
spring.jta.bitronix.datasource.acquisition-interval=1 # Time, in seconds, to wait before trying to acquire a connection again after an invalid connection was acquired.
spring.jta.bitronix.datasource.acquisition-timeout=30 # Timeout, in seconds, for acquiring connections from the pool.
spring.jta.bitronix.datasource.allow-local-transactions=true # Whether or not the transaction manager should allow mixing XA and non-XA transactions.
spring.jta.bitronix.datasource.apply-transaction-timeout=false # Whether or not the transaction timeout should be set on the XAResource when it is enlisted.
spring.jta.bitronix.datasource.automatic-enlisting-enabled=true # Whether or not resources should be enlisted and delisted automatically.
spring.jta.bitronix.datasource.cursor-holdability= # The default cursor holdability for connections.
spring.jta.bitronix.datasource.defer-connection-release=true # Whether or not the database can run many transactions on the same connection and supports transaction interleaving.
spring.jta.bitronix.datasource.enable-jdbc4-connection-test= # Whether or not Connection.isValid() is called when acquiring a connection from the pool.
```

```
spring.jta.bitronix.datasource.ignore-recovery-failures=false # Whether or not
recovery failures should be ignored.
spring.jta.bitronix.datasource.isolation-level= # The default isolation level for
connections.
spring.jta.bitronix.datasource.local-auto-commit= # The default auto-commit mode
for local transactions.
spring.jta.bitronix.datasource.login-timeout= # Timeout, in seconds, for
establishing a database connection.
spring.jta.bitronix.datasource.max-idle-time=60 # The time, in seconds, after
which connections are cleaned up from the pool.
spring.jta.bitronix.datasource.max-pool-size=10 # The maximum size of the pool. 0
denotes no limit.
spring.jta.bitronix.datasource.min-pool-size=0 # The minimum size of the pool.
spring.jta.bitronix.datasource.prepared-statement-cache-size=0 # The target size
of the prepared statement cache. 0 disables the cache.
spring.jta.bitronix.datasource.share-transaction-connections=false #  Whether or
not connections in the ACCESSIBLE state can be shared within the context of a
transaction.
spring.jta.bitronix.datasource.test-query= # SQL query or statement used to
validate a connection before returning it.
spring.jta.bitronix.datasource.two-pc-ordering-position=1 # The position that this
resource should take during two-phase commit (always first is Integer.MIN_VALUE,
always last is Integer.MAX_VALUE).
spring.jta.bitronix.datasource.unique-name=dataSource # The unique name used to
identify the resource during recovery.
spring.jta.bitronix.datasource.use-tm-join=true Whether or not TMJOIN should be
used when starting XAResources.
spring.jta.bitronix.properties.allow-multiple-lrc=false # Allow multiple LRC
resources to be enlisted into the same transaction.
spring.jta.bitronix.properties.asynchronous2-pc=false # Enable asynchronously
execution of two phase commit.
spring.jta.bitronix.properties.background-recovery-interval-seconds=60 # Interval
in seconds at which to run the recovery process in the background.
spring.jta.bitronix.properties.current-node-only-recovery=true # Recover only the
current node.
spring.jta.bitronix.properties.debug-zero-resource-transaction=false # Log the
creation and commit call stacks of transactions executed without a single enlisted
resource.
spring.jta.bitronix.properties.default-transaction-timeout=60 # Default
transaction timeout in seconds.
spring.jta.bitronix.properties.disable-jmx=false # Enable JMX support.
spring.jta.bitronix.properties.exception-analyzer= # Set the fully qualified name
of the exception analyzer implementation to use.
spring.jta.bitronix.properties.filter-log-status=false # Enable filtering of logs
so that only mandatory logs are written.
spring.jta.bitronix.properties.force-batching-enabled=true #  Set if disk forces
are batched.
spring.jta.bitronix.properties.forced-write-enabled=true # Set if logs are forced
to disk.
```

```
spring.jta.bitronix.properties.graceful-shutdown-interval=60 # Maximum amount of
seconds the TM will wait for transactions to get done before aborting them at
shutdown time.
spring.jta.bitronix.properties.jndi-transaction-synchronization-registry-name= #
JNDI name of the TransactionSynchronizationRegistry.
spring.jta.bitronix.properties.jndi-user-transaction-name= # JNDI name of the
UserTransaction.
spring.jta.bitronix.properties.journal=disk # Name of the journal. Can be 'disk',
'null' or a class name.
spring.jta.bitronix.properties.log-part1-filename=btm1.tlog # Name of the first
fragment of the journal.
spring.jta.bitronix.properties.log-part2-filename=btm2.tlog # Name of the second
fragment of the journal.
spring.jta.bitronix.properties.max-log-size-in-mb=2 # Maximum size in megabytes of
the journal fragments.
spring.jta.bitronix.properties.resource-configuration-filename= # ResourceLoader
configuration file name.
spring.jta.bitronix.properties.server-id= # ASCII ID that must uniquely identify
this TM instance. Default to the machine's IP address.
spring.jta.bitronix.properties.skip-corrupted-logs=false # Skip corrupted
transactions log entries.
spring.jta.bitronix.properties.warn-about-zero-resource-transaction=true # Log a
warning for transactions executed without a single enlisted resource.

# NARAYANA (NarayanaProperties)
spring.jta.narayana.default-timeout=60 # Transaction timeout in seconds.
spring.jta.narayana.expiry-
scanners=com.arjuna.ats.internal.arjuna.recovery.ExpiredTransactionStatusManagerSc
anner # 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)
```

```
spring.mongodb.embedded.features=SYNC_DELAY # Comma-separated list of features to
enable.
spring.mongodb.embedded.storage.database-dir= # Directory used for data storage.
spring.mongodb.embedded.storage.oplog-size= # Maximum size of the oplog in
megabytes.
spring.mongodb.embedded.storage.repl-set-name= # Name of the replica set.
spring.mongodb.embedded.version=2.6.10 # Version of Mongo to use.

# REDIS (RedisProperties)
spring.redis.cluster.max-redirects= # Maximum number of redirects to follow when
executing commands across the cluster.
spring.redis.cluster.nodes= # Comma-separated list of "host:port" pairs to
bootstrap from.
spring.redis.database=0 # Database index used by the connection factory.
spring.redis.url= # Connection URL, will override host, port and password (user
will be ignored), e.g. redis://user:password@example.com:6379
spring.redis.host=localhost # Redis server host.
spring.redis.password= # Login password of the redis server.
spring.redis.ssl=false # Enable SSL support.
spring.redis.pool.max-active=8 # Max number of connections that can be allocated
by the pool at a given time. Use a negative value for no limit.
spring.redis.pool.max-idle=8 # Max number of "idle" connections in the pool. Use a
negative value to indicate an unlimited number of idle connections.
spring.redis.pool.max-wait=-1 # Maximum amount of time (in milliseconds) a
connection allocation should block before throwing an exception when the pool is
exhausted. Use a negative value to block indefinitely.
spring.redis.pool.min-idle=0 # Target for the minimum number of idle connections
to maintain in the pool. This setting only has an effect if it is positive.
spring.redis.port=6379 # Redis server port.
spring.redis.sentinel.master= # Name of Redis server.
spring.redis.sentinel.nodes= # Comma-separated list of host:port pairs.
spring.redis.timeout=0 # Connection timeout in milliseconds.

# TRANSACTION (TransactionProperties)
spring.transaction.default-timeout= # Default transaction timeout in seconds.
spring.transaction.rollback-on-commit-failure= # Perform the rollback on commit
failures.



# ----------------------------------------
# INTEGRATION PROPERTIES
# ----------------------------------------

# ACTIVEMQ (ActiveMQProperties)
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)
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)
spring.batch.initializer.enabled= # Create the required batch tables on startup if
necessary. Enabled automatically if no custom table prefix is set or if a custom
schema is configured.
spring.batch.job.enabled=true # Execute all Spring Batch jobs in the context on
startup.
spring.batch.job.names= # Comma-separated list of job names to execute on startup
(For instance `job1,job2`). By default, all Jobs found in the context are
executed.
spring.batch.schema=classpath:org/springframework/batch/core/schema-
@@platform@@.sql # Path to the SQL file to use to initialize the database schema.
spring.batch.table-prefix= # Table prefix for all the batch meta-data tables.

# JMS (JmsProperties)
spring.jms.jndi-name= # Connection factory JNDI name. When set, takes precedence
to others connection factory auto-configurations.
```

```
spring.jms.listener.acknowledge-mode= # Acknowledge mode of the container. By
default, the listener is transacted with automatic acknowledgment.
spring.jms.listener.auto-startup=true # Start the container automatically on
startup.
spring.jms.listener.concurrency= # Minimum number of concurrent consumers.
spring.jms.listener.max-concurrency= # Maximum number of concurrent consumers.
spring.jms.pub-sub-domain=false # Specify if the default destination type is
topic.
spring.jms.template.default-destination= # Default destination to use on
send/receive operations that do not have a destination parameter.
spring.jms.template.delivery-delay= # Delivery delay to use for send calls in
milliseconds.
spring.jms.template.delivery-mode= # Delivery mode. Enable QoS when set.
spring.jms.template.priority= # Priority of a message when sending. Enable QoS
when set.
spring.jms.template.qos-enabled= # Enable explicit QoS when sending a message.
spring.jms.template.receive-timeout= # Timeout to use for receive calls in
milliseconds.
spring.jms.template.time-to-live= # Time-to-live of a message when sending in
milliseconds. Enable QoS when set.

# APACHE KAFKA (KafkaProperties)
spring.kafka.bootstrap-servers= # Comma-delimited list of host:port pairs to use
for establishing the initial connection to the Kafka cluster.
spring.kafka.client-id= # Id to pass to the server when making requests; used for
server-side logging.
spring.kafka.consumer.auto-commit-interval= # Frequency in milliseconds that the
consumer offsets are auto-committed to Kafka if 'enable.auto.commit' true.
spring.kafka.consumer.auto-offset-reset= # What to do when there is no initial
offset in Kafka or if the current offset does not exist any more on the server.
spring.kafka.consumer.bootstrap-servers= # Comma-delimited list of host:port pairs
to use for establishing the initial connection to the Kafka cluster.
spring.kafka.consumer.client-id= # Id to pass to the server when making requests;
used for server-side logging.
spring.kafka.consumer.enable-auto-commit= # If true the consumer's offset will be
periodically committed in the background.
spring.kafka.consumer.fetch-max-wait= # Maximum amount of time in milliseconds the
server will block before answering the fetch request if there isn't sufficient
data to immediately satisfy the requirement given by "fetch.min.bytes".
spring.kafka.consumer.fetch-min-size= # Minimum amount of data the server should
return for a fetch request in bytes.
spring.kafka.consumer.group-id= # Unique string that identifies the consumer group
this consumer belongs to.
spring.kafka.consumer.heartbeat-interval= # Expected time in milliseconds between
heartbeats to the consumer coordinator.
spring.kafka.consumer.key-deserializer= # Deserializer class for keys.
spring.kafka.consumer.max-poll-records= # Maximum number of records returned in a
single call to poll().
spring.kafka.consumer.value-deserializer= # Deserializer class for values.
```

```
spring.kafka.listener.ack-count= # Number of records between offset commits when ackMode is "COUNT" or "COUNT_TIME".
spring.kafka.listener.ack-mode= # Listener AckMode; see the spring-kafka documentation.
spring.kafka.listener.ack-time= # Time in milliseconds between offset commits when ackMode is "TIME" or "COUNT_TIME".
spring.kafka.listener.concurrency= # Number of threads to run in the listener containers.
spring.kafka.listener.poll-timeout= # Timeout in milliseconds to use when polling the consumer.
spring.kafka.producer.acks= # Number of acknowledgments the producer requires the leader to have received before considering a request complete.
spring.kafka.producer.batch-size= # Number of records to batch before sending.
spring.kafka.producer.bootstrap-servers= # Comma-delimited list of host:port pairs to use for establishing the initial connection to the Kafka cluster.
spring.kafka.producer.buffer-memory= # Total bytes of memory the producer can use to buffer records waiting to be sent to the server.
spring.kafka.producer.client-id= # Id to pass to the server when making requests; used for server-side logging.
spring.kafka.producer.compression-type= # Compression type for all data generated by the producer.
spring.kafka.producer.key-serializer= # Serializer class for keys.
spring.kafka.producer.retries= # When greater than zero, enables retrying of failed sends.
spring.kafka.producer.value-serializer= # Serializer class for values.
spring.kafka.properties.*= # Additional properties used to configure the client.
spring.kafka.ssl.key-password= # Password of the private key in the key store file.
spring.kafka.ssl.keystore-location= # Location of the key store file.
spring.kafka.ssl.keystore-password= # Store password for the key store file.
spring.kafka.ssl.truststore-location= # Location of the trust store file.
spring.kafka.ssl.truststore-password= # Store password for the trust store file.
spring.kafka.template.default-topic= # Default topic to which messages will be sent.

# RABBIT (RabbitProperties)
spring.rabbitmq.addresses= # Comma-separated list of addresses to which the client should connect.
spring.rabbitmq.cache.channel.checkout-timeout= # Number of milliseconds to wait to obtain a channel if the cache size has been reached.
spring.rabbitmq.cache.channel.size= # Number of channels to retain in the cache.
spring.rabbitmq.cache.connection.mode=channel # Connection factory cache mode.
spring.rabbitmq.cache.connection.size= # Number of connections to cache.
spring.rabbitmq.connection-timeout= # Connection timeout, in milliseconds; zero for infinite.
spring.rabbitmq.dynamic=true # Create an AmqpAdmin bean.
spring.rabbitmq.host=localhost # RabbitMQ host.
spring.rabbitmq.listener.acknowledge-mode= # Acknowledge mode of container.
```

```
spring.rabbitmq.listener.auto-startup=true # Start the container automatically on
startup.
spring.rabbitmq.listener.concurrency= # Minimum number of consumers.
spring.rabbitmq.listener.default-requeue-rejected= # Whether or not to requeue
delivery failures; default `true`.
spring.rabbitmq.listener.idle-event-interval= # How often idle container events
should be published in milliseconds.
spring.rabbitmq.listener.max-concurrency= # Maximum number of consumers.
spring.rabbitmq.listener.prefetch= # Number of messages to be handled in a single
request. It should be greater than or equal to the transaction size (if used).
spring.rabbitmq.listener.retry.enabled=false # Whether or not publishing retries
are enabled.
spring.rabbitmq.listener.retry.initial-interval=1000 # Interval between the first
and second attempt to deliver a message.
spring.rabbitmq.listener.retry.max-attempts=3 # Maximum number of attempts to
deliver a message.
spring.rabbitmq.listener.retry.max-interval=10000 # Maximum interval between
attempts.
spring.rabbitmq.listener.retry.multiplier=1.0 # A multiplier to apply to the
previous delivery retry interval.
spring.rabbitmq.listener.retry.stateless=true # Whether or not retry is stateless
or stateful.
spring.rabbitmq.listener.transaction-size= # Number of messages to be processed in
a transaction. For best results it should be less than or equal to the prefetch
count.
spring.rabbitmq.password= # Login to authenticate against the broker.
spring.rabbitmq.port=5672 # RabbitMQ port.
spring.rabbitmq.publisher-confirms=false # Enable publisher confirms.
spring.rabbitmq.publisher-returns=false # Enable publisher returns.
spring.rabbitmq.requested-heartbeat= # Requested heartbeat timeout, in seconds;
zero for none.
spring.rabbitmq.ssl.enabled=false # Enable SSL support.
spring.rabbitmq.ssl.key-store= # Path to the key store that holds the SSL
certificate.
spring.rabbitmq.ssl.key-store-password= # Password used to access the key store.
spring.rabbitmq.ssl.trust-store= # Trust store that holds SSL certificates.
spring.rabbitmq.ssl.trust-store-password= # Password used to access the trust
store.
spring.rabbitmq.ssl.algorithm= # SSL algorithm to use. By default configure by the
rabbit client library.
spring.rabbitmq.template.mandatory=false # Enable mandatory messages.
spring.rabbitmq.template.receive-timeout=0 # Timeout for `receive()` methods.
spring.rabbitmq.template.reply-timeout=5000 # Timeout for `sendAndReceive()`
methods.
spring.rabbitmq.template.retry.enabled=false # Set to true to enable retries in
the `RabbitTemplate`.
spring.rabbitmq.template.retry.initial-interval=1000 # Interval between the first
and second attempt to publish a message.
```

```
spring.rabbitmq.template.retry.max-attempts=3 # Maximum number of attempts to
publish a message.
spring.rabbitmq.template.retry.max-interval=10000 # Maximum number of attempts to
publish a message.
spring.rabbitmq.template.retry.multiplier=1.0 # A multiplier to apply to the
previous publishing retry interval.
spring.rabbitmq.username= # Login user to authenticate to the broker.
spring.rabbitmq.virtual-host= # Virtual host to use when connecting to the broker.


# ----------------------------------------
# ACTUATOR PROPERTIES
# ----------------------------------------

# ENDPOINTS (AbstractEndpoint subclasses)
endpoints.enabled=true # Enable endpoints.
endpoints.sensitive= # Default endpoint sensitive setting.
endpoints.actuator.enabled=true # Enable the endpoint.
endpoints.actuator.path= # Endpoint URL path.
endpoints.actuator.sensitive=false # Enable security on the endpoint.
endpoints.auditevents.enabled= # Enable the endpoint.
endpoints.auditevents.path= # Endpoint path.
endpoints.auditevents.sensitive=false # Enable security on the endpoint.
endpoints.autoconfig.enabled= # Enable the endpoint.
endpoints.autoconfig.id= # Endpoint identifier.
endpoints.autoconfig.path= # Endpoint path.
endpoints.autoconfig.sensitive= # Mark if the endpoint exposes sensitive
information.
endpoints.beans.enabled= # Enable the endpoint.
endpoints.beans.id= # Endpoint identifier.
endpoints.beans.path= # Endpoint path.
endpoints.beans.sensitive= # Mark if the endpoint exposes sensitive information.
endpoints.configprops.enabled= # Enable the endpoint.
endpoints.configprops.id= # Endpoint identifier.
endpoints.configprops.keys-to-
sanitize=password,secret,key,token,.*credentials.*,vcap_services # Keys that
should be sanitized. Keys can be simple strings that the property ends with or
regex expressions.
endpoints.configprops.path= # Endpoint path.
endpoints.configprops.sensitive= # Mark if the endpoint exposes sensitive
information.
endpoints.docs.curies.enabled=false # Enable the curie generation.
endpoints.docs.enabled=true # Enable actuator docs endpoint.
endpoints.docs.path=/docs #
endpoints.docs.sensitive=false #
endpoints.dump.enabled= # Enable the endpoint.
endpoints.dump.id= # Endpoint identifier.
endpoints.dump.path= # Endpoint path.
endpoints.dump.sensitive= # Mark if the endpoint exposes sensitive information.
```

```
endpoints.env.enabled= # Enable the endpoint.
endpoints.env.id= # Endpoint identifier.
endpoints.env.keys-to-
sanitize=password,secret,key,token,.*credentials.*,vcap_services # Keys that
should be sanitized. Keys can be simple strings that the property ends with or
regex expressions.
endpoints.env.path= # Endpoint path.
endpoints.env.sensitive= # Mark if the endpoint exposes sensitive information.
endpoints.flyway.enabled= # Enable the endpoint.
endpoints.flyway.id= # Endpoint identifier.
endpoints.flyway.sensitive= # Mark if the endpoint exposes sensitive information.
endpoints.health.enabled= # Enable the endpoint.
endpoints.health.id= # Endpoint identifier.
endpoints.health.mapping.*= # Mapping of health statuses to HttpStatus codes. By
default, registered health statuses map to sensible defaults (i.e. UP maps to
200).
endpoints.health.path= # Endpoint path.
endpoints.health.sensitive= # Mark if the endpoint exposes sensitive information.
endpoints.health.time-to-live=1000 # Time to live for cached result, in
milliseconds.
endpoints.heapdump.enabled= # Enable the endpoint.
endpoints.heapdump.path= # Endpoint path.
endpoints.heapdump.sensitive= # Mark if the endpoint exposes sensitive
information.
endpoints.hypermedia.enabled=false # Enable hypermedia support for endpoints.
endpoints.info.enabled= # Enable the endpoint.
endpoints.info.id= # Endpoint identifier.
endpoints.info.path= # Endpoint path.
endpoints.info.sensitive= # Mark if the endpoint exposes sensitive information.
endpoints.jolokia.enabled=true # Enable Jolokia endpoint.
endpoints.jolokia.path=/jolokia # Endpoint URL path.
endpoints.jolokia.sensitive=true # Enable security on the endpoint.
endpoints.liquibase.enabled= # Enable the endpoint.
endpoints.liquibase.id= # Endpoint identifier.
endpoints.liquibase.sensitive= # Mark if the endpoint exposes sensitive
information.
endpoints.logfile.enabled=true # Enable the endpoint.
endpoints.logfile.external-file= # External Logfile to be accessed.
endpoints.logfile.path=/logfile # Endpoint URL path.
endpoints.logfile.sensitive=true # Enable security on the endpoint.
endpoints.loggers.enabled=true # Enable the endpoint.
endpoints.loggers.id= # Endpoint identifier.
endpoints.loggers.path=/logfile # Endpoint path.
endpoints.loggers.sensitive=true # Mark if the endpoint exposes sensitive
information.
endpoints.mappings.enabled= # Enable the endpoint.
endpoints.mappings.id= # Endpoint identifier.
endpoints.mappings.path= # Endpoint path.
```

```
endpoints.mappings.sensitive= # Mark if the endpoint exposes sensitive
information.
endpoints.metrics.enabled= # Enable the endpoint.
endpoints.metrics.filter.enabled=true # Enable the metrics servlet filter.
endpoints.metrics.filter.gauge-submissions=merged # Http filter gauge submissions
(merged, per-http-method)
endpoints.metrics.filter.counter-submissions=merged # Http filter counter
submissions (merged, per-http-method)
endpoints.metrics.id= # Endpoint identifier.
endpoints.metrics.path= # Endpoint path.
endpoints.metrics.sensitive= # Mark if the endpoint exposes sensitive information.
endpoints.shutdown.enabled= # Enable the endpoint.
endpoints.shutdown.id= # Endpoint identifier.
endpoints.shutdown.path= # Endpoint path.
endpoints.shutdown.sensitive= # Mark if the endpoint exposes sensitive
information.
endpoints.trace.enabled= # Enable the endpoint.
endpoints.trace.id= # Endpoint identifier.
endpoints.trace.path= # Endpoint path.
endpoints.trace.sensitive= # Mark if the endpoint exposes sensitive information.

# ENDPOINTS CORS CONFIGURATION (EndpointCorsProperties)
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)
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)
jolokia.config.*= # See Jolokia manual

# MANAGEMENT HTTP SERVER (ManagementServerProperties)
```

```
management.add-application-context-header=true # Add the "X-Application-Context"
HTTP header in each response.
management.address= # Network address that the management endpoints should bind
to.
management.context-path= # Management endpoint context-path. For instance
`/actuator`
management.cloudfoundry.enabled= # Enable extended Cloud Foundry actuator
endpoints
management.cloudfoundry.skip-ssl-validation= # Skip SSL verification for Cloud
Foundry actuator endpoint security calls
management.port= # Management endpoint HTTP port. Uses the same port as the
application by default. Configure a different port to use management-specific SSL.
management.security.enabled=true # Enable security.
management.security.roles=ACTUATOR # Comma-separated list of roles that can access
the management endpoint.
management.security.sessions=stateless # Session creating policy to use (always,
never, if_required, stateless).
management.ssl.ciphers= # Supported SSL ciphers. Requires a custom
management.port.
management.ssl.client-auth= # Whether client authentication is wanted ("want") or
needed ("need"). Requires a trust store. Requires a custom management.port.
management.ssl.enabled= # Enable SSL support. Requires a custom management.port.
management.ssl.enabled-protocols= # Enabled SSL protocols. Requires a custom
management.port.
management.ssl.key-alias= # Alias that identifies the key in the key store.
Requires a custom management.port.
management.ssl.key-password= # Password used to access the key in the key store.
Requires a custom management.port.
management.ssl.key-store= # Path to the key store that holds the SSL certificate
(typically a jks file). Requires a custom management.port.
management.ssl.key-store-password= # Password used to access the key store.
Requires a custom management.port.
management.ssl.key-store-provider= # Provider for the key store. Requires a custom
management.port.
management.ssl.key-store-type= # Type of the key store. Requires a custom
management.port.
management.ssl.protocol=TLS # SSL protocol to use. Requires a custom
management.port.
management.ssl.trust-store= # Trust store that holds SSL certificates. Requires a
custom management.port.
management.ssl.trust-store-password= # Password used to access the trust store.
Requires a custom management.port.
management.ssl.trust-store-provider= # Provider for the trust store. Requires a
custom management.port.
management.ssl.trust-store-type= # Type of the trust store. Requires a custom
management.port.

# HEALTH INDICATORS
management.health.db.enabled=true # Enable database health check.
```

```
management.health.cassandra.enabled=true # Enable cassandra health check.
management.health.couchbase.enabled=true # Enable couchbase health check.
management.health.defaults.enabled=true # Enable default health indicators.
management.health.diskspace.enabled=true # Enable disk space health check.
management.health.diskspace.path= # Path used to compute the available disk space.
management.health.diskspace.threshold=0 # Minimum disk space that should be
available, in bytes.
management.health.elasticsearch.enabled=true # Enable elasticsearch health check.
management.health.elasticsearch.indices= # Comma-separated index names.
management.health.elasticsearch.response-timeout=100 # The time, in milliseconds,
to wait for a response from the cluster.
management.health.jms.enabled=true # Enable JMS health check.
management.health.ldap.enabled=true # Enable LDAP health check.
management.health.mail.enabled=true # Enable Mail health check.
management.health.mongo.enabled=true # Enable MongoDB health check.
management.health.rabbit.enabled=true # Enable RabbitMQ health check.
management.health.redis.enabled=true # Enable Redis health check.
management.health.solr.enabled=true # Enable Solr health check.
management.health.status.order=DOWN, OUT_OF_SERVICE, UP, UNKNOWN # Comma-separated
list of health statuses in order of severity.

# INFO CONTRIBUTORS (InfoContributorProperties)
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)
management.shell.auth.type=simple # Authentication type. Auto-detected according
to the environment.
management.shell.auth.jaas.domain=my-domain # JAAS domain.
management.shell.auth.key.path= # Path to the authentication key. This should
point to a valid ".pem" file.
management.shell.auth.simple.user.name=user # Login user.
management.shell.auth.simple.user.password= # Login password.
management.shell.auth.spring.roles=ACTUATOR # Comma-separated list of required
roles to login to the CRaSH console.
management.shell.command-path-
patterns=classpath*:/commands/**,classpath*:/crash/commands/** # Patterns to use
to look for commands.
management.shell.command-refresh-interval=-1 # Scan for changes and update the
command if necessary (in seconds).
management.shell.config-path-patterns=classpath*:/crash/* # Patterns to use to
look for configurations.
management.shell.disabled-commands=jpa*,jdbc*,jndi* # Comma-separated list of
commands to disable.
management.shell.disabled-plugins= # Comma-separated list of plugins to disable.
Certain plugins are disabled by default based on the environment.
```

```
management.shell.ssh.auth-timeout = # Number of milliseconds after user will be
prompted to login again.
management.shell.ssh.enabled=true # Enable CRaSH SSH support.
management.shell.ssh.idle-timeout = # Number of milliseconds after which unused
connections are closed.
management.shell.ssh.key-path= # Path to the SSH server key.
management.shell.ssh.port=2000 # SSH port.
management.shell.telnet.enabled=false # Enable CRaSH telnet support. Enabled by
default if the TelnetPlugin is  available.
management.shell.telnet.port=5000 # Telnet port.

# TRACING (TraceProperties)
management.trace.include=request-headers,response-headers,cookies,errors # Items
to be included in the trace.

# METRICS EXPORT (MetricExportProperties)
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)
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)
spring.devtools.remote.context-path=/.~~spring-boot!~ # Context path used to
handle the remote connection.
spring.devtools.remote.debug.enabled=true # Enable remote debug support.
spring.devtools.remote.debug.local-port=8000 # Local remote debug server port.
spring.devtools.remote.proxy.host= # The host of the proxy to use to connect to
the remote application.
spring.devtools.remote.proxy.port= # The port of the proxy to use to connect to
the remote application.
spring.devtools.remote.restart.enabled=true # Enable remote restart.
spring.devtools.remote.secret= # A shared secret required to establish a
connection (required to enable remote support).
spring.devtools.remote.secret-header-name=X-AUTH-TOKEN # HTTP header used to
transfer the shared secret.


# ----------------------------------------
# TESTING PROPERTIES
# ----------------------------------------

spring.test.database.replace=any # Type of existing DataSource to replace.
spring.test.mockmvc.print=default # MVC Print option.
```

附录2 sspring.facyories

```
# PropertySource Loaders
org.springframework.boot.env.PropertySourceLoader=\
org.springframework.boot.env.PropertiesPropertySourceLoader,\
org.springframework.boot.env.YamlPropertySourceLoader

# Run Listeners
```

```
org.springframework.boot.SpringApplicationRunListener=\
org.springframework.boot.context.event.EventPublishingRunListener

# Application Context Initializers
org.springframework.context.ApplicationContextInitializer=\
org.springframework.boot.context.ConfigurationWarningsApplicationContextInitialize
r,\
org.springframework.boot.context.ContextIdApplicationContextInitializer,\
org.springframework.boot.context.config.DelegatingApplicationContextInitializer,\
org.springframework.boot.context.embedded.ServerPortInfoApplicationContextInitiali
zer

# Application Listeners
org.springframework.context.ApplicationListener=\
org.springframework.boot.ClearCachesApplicationListener,\
org.springframework.boot.builder.ParentContextCloserApplicationListener,\
org.springframework.boot.context.FileEncodingApplicationListener,\
org.springframework.boot.context.config.AnsiOutputApplicationListener,\
org.springframework.boot.context.config.ConfigFileApplicationListener,\
org.springframework.boot.context.config.DelegatingApplicationListener,\
org.springframework.boot.liquibase.LiquibaseServiceLocatorApplicationListener,\
org.springframework.boot.logging.ClasspathLoggingApplicationListener,\
org.springframework.boot.logging.LoggingApplicationListener

# Environment Post Processors
org.springframework.boot.env.EnvironmentPostProcessor=\
org.springframework.boot.cloud.CloudFoundryVcapEnvironmentPostProcessor,\
org.springframework.boot.env.SpringApplicationJsonEnvironmentPostProcessor

# Failure Analyzers
org.springframework.boot.diagnostics.FailureAnalyzer=\
org.springframework.boot.diagnostics.analyzer.BeanCurrentlyInCreationFailureAnalyz
er,\
org.springframework.boot.diagnostics.analyzer.BeanNotOfRequiredTypeFailureAnalyzer
,\
org.springframework.boot.diagnostics.analyzer.BindFailureAnalyzer,\
org.springframework.boot.diagnostics.analyzer.ConnectorStartFailureAnalyzer,\
org.springframework.boot.diagnostics.analyzer.NoUniqueBeanDefinitionFailureAnalyze
r,\
org.springframework.boot.diagnostics.analyzer.PortInUseFailureAnalyzer,\
org.springframework.boot.diagnostics.analyzer.ValidationExceptionFailureAnalyzer

# FailureAnalysisReporters
org.springframework.boot.diagnostics.FailureAnalysisReporter=\
org.springframework.boot.diagnostics.LoggingFailureAnalysisReporter
```