**Spring Boot**

# Spring Boot 简介

Spring 诞生时是 Java 企业版（Java Enterprise Edition，JEE，也称 J2EE）的轻量级代替品。无需开发重量级的 Enterprise JavaBean（EJB），Spring 为企业级Java 开发提供了一种相对简单的方法，通过依赖注入和面向切面编程，用简单的Java 对象（Plain Old Java Object，POJO）实现了 EJB 的功能。

## 发展背景

*虽然 Spring 的组件代码是轻量级的，但它的配置却是重量级的。*

**第一阶段：xml配置**

在Spring 1.x时代，使用Spring开发满眼都是xml配置的Bean，随着项目的扩大，我们需要把xml配置文件放到不同的配置文件里，那时需要频繁的在开发的类和配置文件之间进行切换

**第二阶段：注解配置**

在Spring 2.x 时代，随着JDK1.5带来的注解支持，Spring提供了声明Bean的注解（例如@Component、@Service），大大减少了配置量。主要使用的方式是应用的基本配置（如数据库配置）用xml，业务配置用注解

**第三阶段：java配置**

Spring 3.0 引入了基于 Java 的配置能力，这是一种类型安全的可重构配置方式，可以代替 XML。*我们目前刚好处于这个时代，Spring4.x和Spring Boot都推荐使用Java配置*。

***所有这些配置都代表了开发时的损耗***。

因为在思考 Spring 特性配置和解决业务问题之间需要进行思维切换，所以写配置挤占了写应用程序逻辑的时间。

除此之外，项目的依赖管理也是件吃力不讨好的事情。决定项目里要用哪些库就已经够让人头痛的了，你还要知道这些库的哪个版本和其他库不会有冲突，这难题实在太棘手。并且，依赖管理也是一种损耗，添加依赖不是写应用程序代码。一旦选错了依赖的版本，随之而来的不兼容问题毫无疑问会是生产力杀手。

Spring Boot 让这一切成为了过去。

Spring Boot 简化了基于Spring的应用开发，只需要“run”就能创建一个独立的、生产级别的Spring应用。

Spring Boot为Spring平台及第三方库提供开箱即用的设置（提供默认设置），这样我们就可以简单的开始。多数Spring Boot应用只需要很少的Spring配置。

我们可以使用SpringBoot创建java应用，并使用java –jar 启动它，或者采用传统的war部署方式。

## 主要目标

* 为所有 Spring 的开发提供一个从根本上更快的入门体验
* 开箱即用，但通过自己设置参数，即可快速摆脱这种方式。
* 提供了一些大型项目中常见的非功能性特性，如内嵌服务器、安全、指标，健康检测、外部化配置等
* 绝对没有代码生成，也无需 XML 配置。

## 核心概念

### 起步依赖

要理解Spring Boot起步依赖带来的好处，先让我们假设它们尚不存在。如果没用Spring Boot的话，你会向项目里添加哪些依赖呢？要用Spring MVC的话，你需要哪些依赖？

你自己脑海里同时想起了很多要添加的依赖，但是我不敢保证的是，你是否有漏掉的依赖？

不过不管怎么做，看来如果没有Spring Boot起步依赖，你就有不少功课要做。

Spring Boot通过提供众多起步依赖降低项目依赖的复杂度。

起步依赖本质上是一个Maven项目对象模型（Project Object Model，POM），定义了对其他库的传递依赖，这些东西加在一起即支持某项功能。***很多起步依赖的命名都暗示了它们提供的某种或某类功能。***

举例来说，你打算搭建一个项目做一个Web应用程序。与其向项目的构建文件

里添加一堆单独的库依赖，还不如声明这是一个Web应用程序来得简单。你只要添加Spring Boot的Web起步依赖（spring-boot-starter-web）就好了。

### 自动配置

spring boot除了强大的起步依赖之外，还有一个强大概念就是自动配置。

简而言之，Spring Boot的自动配置是一个运行时（更准确地说，是应用程序启动时）的过程，考虑了众多因素，才决定Spring配置应该用哪个，不该用哪个。

举几个例子，下面这些情况都是Spring Boot的自动配置要考虑的。

* Spring的DataSourceTransactionManager是不是在Classpath里？如果是，并且有DataSource的Bean，则自动配置一个DataSourceTransactionManager的Bean。
* Spring的SqlSessionFactory是不是在Classpath里？如果是，并且有DataSource的Bean，则自动配置一个SqlSessionFactory的Bean。

每当应用程序启动的时候，Spring Boot的自动配置都要做将近200个这样的决定，涵盖安全、集成、持久化、Web开发等诸多方面。所有这些自动配置就是为了尽量不让你自己写配置。

# Spring Boot 入门

## 环境准备

IDE：Eclipse Mars2

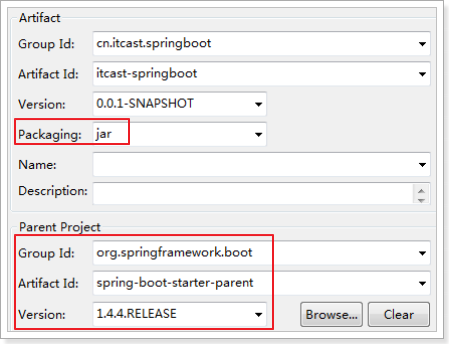
Spring-Boot：1.4.4

Maven： 3.3.3 （官方声明Springboot 1.4.4版本需要Maven 3.2+）

本地仓库：需要使用资料中的仓库

## 工程搭建

### 创建Maven工程



### 添加起步依赖

在pom.xml中添加依赖，效果如下

<parent>

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

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

<version>1.4.4.RELEASE</version>

</parent>

<groupId>cn.itcast.springboot</groupId>

<artifactId>itcast-springboot</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<dependency>

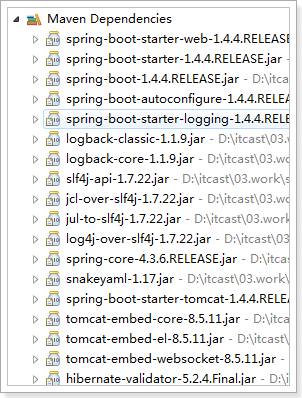
<groupId>org.springframework.boot</groupId>

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

</dependency>

</dependencies>

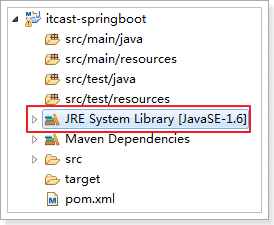
我们会惊奇地发现，我们的工程自动添加了好多好多jar 包，而这些jar 包正式我们做开发时需要导入的jar 包。



因为这些jar 包被我们刚才加入的spring-boot-starter-web 所引用了，所以添加spring-boot-starter-web后会自动把依赖传递过来。

### 变更JDK版本

我们发现默认情况下工程的JDK版本是1.6，但是通常使用的是1.7的版本



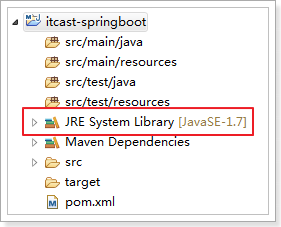
修改JDK为1.7，需要在pom.xml中添加以下配置：

<properties>

<java.version>1.7</java.version>

</properties>

使用Maven更新工程后，就发现版本已经变成1.7了



注意：

虽然JDK1.6或者1.7都可以使用Spring-Boot，但Spring-Boot官方建议使用JDK1.8。要使用JDK1.8，首先必须要配置JDK1.8后，才可以使用上述方法设置。

## 启动引导类

需要创建一个启动引导类：

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

**public** **class** Application {

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(Application.**class**, args);

}

}

@SpringBootApplication***开启了Spring的组件扫描和Spring Boot的自动配置功能***。实际上，@SpringBootApplication将三个有用的注解组合在了一起。

* Spring的@Configuration：***标明该类使用Spring基于Java的配置***。虽然本书不会写太多配置，但我们会更倾向于使用基于Java而不是XML的配置。
* Spring的@ComponentScan：***启用组件扫描***，这样你写的Web控制器类和其他组件才能被自动发现并注册为Spring应用程序上下文里的Bean。本章稍后会写一个简单的Spring MVC控制器，使用@Controller进行注解，这样组件扫描才能找到它。
* Spring Boot 的@EnableAutoConfiguration ： 这个不起眼的小注解就是开启了Spring Boot自动配置的魔力，让你不用再写成篇的配置了。

在Spring Boot的早期版本中，你需要在Application类上同时标上这三个注解，但从Spring Boot 1.2.0开始，有@SpringBootApplication就行了。

## 入门程序

需求：使用Spring MVC实现Hello World输出

### 原来的实现

我们现在开始使用spring MVC 框架，实现json 数据的输出。如果按照我们原来的做法，需要在web.xml 中添加一个DispatcherServlet 的配置，还需要添加一个spring的配置文件，配置文件如下配置

spring加入配置

<!-- controller注解扫描 -->

<context:component-scan base-package=*"cn.itcast.springboot.controller"* />

<!-- 注解驱动 -->

<mvc:annotation-driven />

web.xml加入配置

<!-- 配置前端控制器 -->

<servlet>

<servlet-name> itcast-springboot</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<init-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath:spring/\*.xml</param-value>

</init-param>

</servlet>

<servlet-mapping>

<servlet-name> itcast-springboot</servlet-name>

<url-pattern>/</url-pattern>

</servlet-mapping>

还要编写Controller。。。

### Spring Boot的实现

我们不需要配置文件，直接编写Controller类即可

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RestController;

@Controller

**public** **class** HelloWorldController {

@RequestMapping("info")

@ResponseBody

**public** String info() {

**return** "Hello world!";

}

}

启动方式一：启动之前编写的引导类即可

启动方式二：使用Maven命令spring-boot:run执行即可

选择 Maven Build



在浏览器地址栏输入http://localhost:8080/info 即可看到运行结果

# Spring Boot 实战

## 需求

查询数据库中的用户信息，并且使用easyui展示查询结果。

要求使用的技术：spring boot + spring mvc + mybatis + easyui

## 环境准备

### 导入数据库表

在MySQL数据库执行以下语句

DROP TABLE IF EXISTS `user`;

CREATE TABLE `user` (

`id` bigint(20) NOT NULL AUTO\_INCREMENT,

`username` varchar(100) DEFAULT NULL COMMENT '用户名',

`password` varchar(100) DEFAULT NULL COMMENT '密码',

`name` varchar(100) DEFAULT NULL COMMENT '姓名',

PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=7 DEFAULT CHARSET=utf8;

INSERT INTO `user` VALUES ('1', 'zhangsan', '123456', '张三');

INSERT INTO `user` VALUES ('2', 'lisi', '123456', '李四');

INSERT INTO `user` VALUES ('3', 'wangwu', '123456', '王五');

INSERT INTO `user` VALUES ('4', 'zhangwei', '123456', '张伟');

INSERT INTO `user` VALUES ('5', 'lina', '123456', '李娜');

INSERT INTO `user` VALUES ('6', 'lilei', '123456', '李磊');

## 添加起步依赖

在pom.xml中加入如下mybatis的起步依赖

<!-- MyBatis启动器 -->

<dependency>

<groupId>org.mybatis.spring.boot</groupId>

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

<version>1.1.1</version>

</dependency>

<!-- MySQL连接驱动 -->

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

</dependency>

## 加入配置文件

在src/main/resources 下添加application.properties 配置文件，内容如下：

#DB Configuration:

spring.datasource.driverClassName=com.mysql.jdbc.Driver

spring.datasource.url=jdbc:mysql://127.0.0.1:3306/springboot?useUnicode=true&characterEncoding=utf8

spring.datasource.username=root

spring.datasource.password=root

## 代码实现

### 功能分析

请求URL：user/lis

请求返回值：json数组，数据是用户信息的数据

### 创建pojo

**public** **class** User {

// 主键

**private** Long id;

// 用户名

**private** String username;

// 密码

**private** String password;

// 姓名

**private** String name;

get/set。。。。。。

}

### 编写Mapper

#### Mapper接口

mapper接口类上面必须添加mapper注解。

***@Mapper***：标记该类是一个mybatis的mapper接口，可以被spring boot自动扫描到spring上下文中。

@Mapper

**public** **interface** UserMapper {

**public** List<User> queryUserList();

}

#### Mapper映射文件

在src\main\resources\mapper路径下加入UserMapper.xml配置文件

<?xml version=*"1.0"* encoding=*"UTF-8"* ?>

<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd" >

<mapper namespace=*"cn.itcast.springboot.mapper.UserMapper"*>

<select id=*"queryUserList"* resultType=*"user"*>

select \* from user

</select>

</mapper>

#### 修改配置文件

在application.properties添加配置

#spring集成Mybatis环境

#pojo别名扫描包

mybatis.type-aliases-package=cn.itcast.springboot.pojo

#加载Mybatis映射文件

mybatis.mapper-locations=classpath:mapper/\*Mapper.xml

### 编写Service

编写Service实现类

@Service

**public** **class** UserServiceImpl **implements** UserService {

@Autowired

**private** UserMapper userMapper;

@Override

**public** List<User> queryUserList() {

List<User> list = **this**.userMapper.queryUserList();

**return** list;

}

}

### 编写Controller

编写Controller

@Controller

@RequestMapping("user")

**public** **class** UserControlelr {

@Autowired

**private** UserService userService;

@RequestMapping("list")

@ResponseBody

**public** List<User> list() {

List<User> list = **this**.userService. queryUserList();

**return** list;

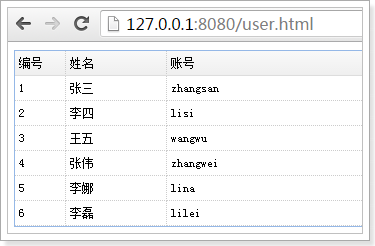
}

}

### 测试

浏览器地址栏输入：http://localhost:8080/user.html

显示效果：



# Spring Boot 整合

## 整合MyBatis分页助手

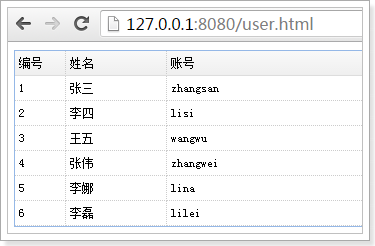
以上所有的配置都是使用的默认配置，我们只需要专注java代码的开发即可，不需要加入spring配置文件了。

但并不是所有得场景都是简单的业务，有时候业务复杂，需要我们加入自定义的配置文件；有时候需要载入例如分页助手这样的插件，辅助开发，所以我们也需要了解如何加载这些配置。

需求：

1. 实现用户分页查询接口(分页助手实现)；
2. 使用Spring Boot + Spring MVC + MyBatis 实现用户列表查询，

效果如下：



### 加入依赖

我们需要加入分页插件，所以需要在pom.xml加入以下依赖

<!-- 分页助手 -->

<dependency>

<groupId>com.github.pagehelper</groupId>

<artifactId>pagehelper</artifactId>

<version>3.7.5</version>

</dependency>

### 功能分析

请求URL：/user/list/{page}/{rows}

请求参数：Integer page,Integer rows

请求返回值：json数组

业务逻辑分析：

1. 根据page和rows，调用PageHelper分页插件，实现分页查询
2. 将查询结果list返回

### 编写Mapper

@Mapper

**public** **interface** UserMapper{

**public** List<User> queryUserList();

}

#### 修改application配置文件

在application.properties添加配置

#加载Mybatis全局配置文件（一般在使用分页助手是再指定）

mybatis.config-location=classpath:mybatis/SqlMapConfig.xml

#### 全局配置文件

在src\main\resources\mybatis加入SqlMapConfig.xml配置文件，用以加载分页助手

<?xml version=*"1.0"* encoding=*"UTF-8"* ?>

<!DOCTYPE configuration

PUBLIC "-//mybatis.org//DTD Config 3.0//EN"

"http://mybatis.org/dtd/mybatis-3-config.dtd">

<configuration>

<!-- 分页助手 -->

<plugins>

<plugin interceptor=*"com.github.pagehelper.PageHelper"*>

<property name=*"dialect"* value=*"mysql"* />

</plugin>

</plugins>

</configuration>

### 编写Service

编写Service接口

**public** **interface** UserService {

List<User> queryUserList();

List<User> queryUserByPage(Integer page, Integer rows);

}

Service实现类编写

@Service

**public** **class** UserServiceImpl **implements** UserService {

@Autowired

**private** UserMapper userMapper;

@Override

**public** List<User> queryUserList() {

List<User> list = **this**.userMapper.queryUserList();

**return** list;

}

//分页助手

@Override

**public** List<User> queryUserByPage(Integer page, Integer rows) {

// 设置分页

PageHelper.*startPage*(page, rows);

List<User> list = **this**.userMapper.queryUserList();

**return** list;

}

}

### 编写Controller

Controller编写

@Controller

@RequestMapping("user")

**public** **class** UserControlelr {

@Autowired

**private** UserService userService;

@RequestMapping("list")

@ResponseBody

**public** List<User> queryUserList() {

List<User> list = **this**.userService.queryUserList();

**return** list;

}

@RequestMapping("list/{page}/{rows}")

@ResponseBody

**public** List<User> queryUserByPage(@PathVariable Integer page, @PathVariable Integer rows) {

List<User> list = **this**.userService.queryUserByPage(page, rows);

**return** list;

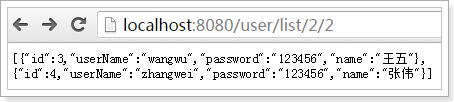
}

}

### 测试

测试使用分页助手

浏览器地址栏输入：http://127.0.0.1:8080/user/list/2/2

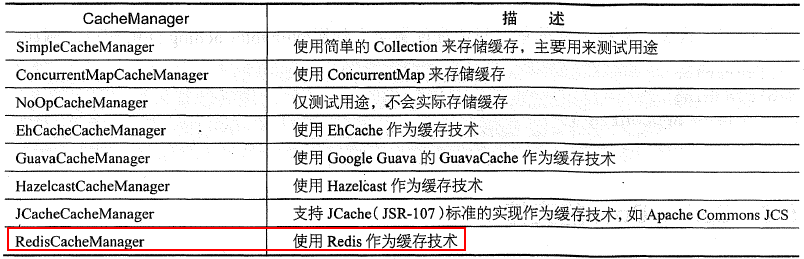


## 整合Redis

### spring cache支持

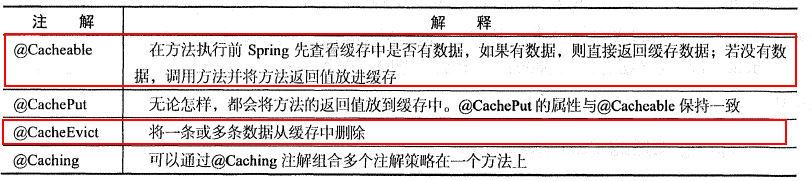
spring 定义了org.springframework.cache.CacheManager和org.springframework.cache.Cache接口用来统一不同的缓存技术。其中，CacheManager是Spring提供的各类缓存技术的抽象接口，Cache接口一般我们不需要打交道。

针对不同的缓存技术，需要实现不同的CacheManager，Spring定义了如下CacheManager实现：



我们在使用CacheManager的时候，需要注册实现的CacheManager的bean，而springboot通过spring-boot-starter-redis起步依赖，为我们***自动配置了RedisCacheManager***以及RedisTemplate的Bean。

spring cache新增了4个注解用来操作缓存：



@Cacheable：添加/使用缓存

@CacheEvict：删除缓存

这两个注解都包含三个属性：value、key、condition。

* 参数value是缓存的名字，在执行的时候，会找叫这个名字的缓存使用/删除
* 参数key是缓存中的key，默认情况下是空串””,表示key的值就是使用@Cacheable注解的方法签名。其中的值可以编写Spring的一种表达式语言SpEL来指定，我们这里也可以随意指定，但是需要注意一定要加单引号。
* 参数condition表示满足什么条件，才能添加或者删除缓存。

### 注解方式实现添加缓存

需求：基于上例代码，将列表数据缓存到Redis

#### 加入依赖

在pom.xml加入依赖

<!-- 配置使用redis启动器 -->

<dependency>

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

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

</dependency>

#### 修改引导类

修改开启缓存，添加注解**@EnableCaching**

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

**import** org.springframework.cache.annotation.EnableCaching;

@SpringBootApplication

**@EnableCaching**

**public** **class** Application {

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(Application.**class**, args);

}

}

#### 设置实现序列化接口

要缓存到redis中的实体，需要让实体实现序列化接口

**public** **class** User **implements** Serializable {

**private** Long id;

**private** String username;

**private** String password;

**private** String name;

。。。。。。

}

#### 实现添加/删除缓存

修改UserServiceImpl，

添加**spring的@Cacheable**注解实现缓存添加

添加**spring的@CacheEvict**注解实现缓存删除

@Override

@CacheEvict(value = "userCache", key = "'user.queryAll'")

**public** List<User> queryUserByName(String name) {

System.***out***.println("缓存清理了！");

List<User> list = **this**.userMapper.queryUserByName(name);

**return** list;

}

// 调用使用UserMapper.xml的Mapper

@Override

@Cacheable(value = "userCache", key = "'user.queryAll'")

**public** List<User> queryAll() {

System.***out***.println("从MySQL中查询");

List<User> list = **this**.userMapper.queryAll();

**return** list;

}

这样设置完成后，执行queryAll()方法就会使用缓存，如果缓存没有就添加缓存，而queryUserByName(String name)方法则是删除缓存

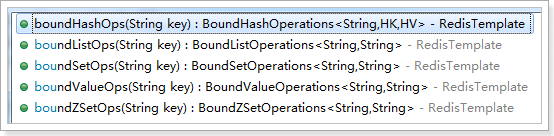
### redis的深入使用

#### 直接操作redis

redis除了作为缓存使用，还有很多其他的作用，例如利用redis的单线程获取唯一数，例如使用redis为单点登录系统存储用户登录信息等，我们就需要直接操作redis。

官网提供了三种接口RedisConnectionFactory, StringRedisTemplate 和 RedisTemplate，我们可以直接注入或者自己实现其他的实现类，来直接操作redis。我们这里使用RedisTemplate来操作Redis。

如下所示，我们只需要直接注入RedisTemplate即可使用以下方法操作redis的五种不同的数据类型



测试：

@Autowired

**private** RedisTemplate<String, String> redisTemplate;

@Override

@CacheEvict(value = "userCache", key = "'user.findAll'")

**public** List<User> queryUserByName(String name) {

// 保存数据

**this**.redisTemplate.boundValueOps("redis").set("Hello redis !");

// 设置有效时间为100秒

**this**.redisTemplate.boundValueOps("redis").expire(100l, TimeUnit.***SECONDS***);

// 给value每次执行加一操作

**this**.redisTemplate.boundValueOps("count").increment(1l);

System.***out***.println("缓存清理了！");

List<User> list = **this**.userMapper.queryUserByName(name);

**return** list;

}

#### 设置redis连接属性

**redis单机版**

redis启动器默认情况下会找本地的redis服务，端口号默认是6379如果需要访问其他服务器的redis，则需要在application.properties中进行如下配置：

#Redis

spring.redis.host=192.168.37.161

spring.redis.port=6379

这表示会去找ip为192.168.37.161和端口为6379的服务

**redis集群版**

#Redis

#spring.redis.host=192.168.37.161

#spring.redis.port=6379

#Redis Cluster

spring.redis.cluster.nodes=192.168.37.161:7001,192.168.37.161:7002,192.168.37.161:7003,192.168.37.161:7004,192.168.37.161:7005,192.168.37.161:7006

切换到集群版只需要做以上配置，配置集群版节点信息，注释掉单机版信息

## 整合ActiveMQ

### 加入依赖

在pom.xml中加入以下配置

<!-- 配置ActiveMQ启动器 -->

<dependency>

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

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

</dependency>

### 创建队列

在引导类中添加以下方法，设置队列

@SpringBootApplication

@EnableCaching

**public** **class** Application {

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(Application.**class**, args);

}

@Bean

**public** Queue queue() {

**return** **new** ActiveMQQueue("itcast.queue");

}

}

### 发送消息

编写Controller，发送消息

**import** javax.jms.Destination;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.jms.core.JmsTemplate;

**import** org.springframework.web.bind.annotation.PathVariable;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RestController;

@RestController

@RequestMapping("queue")

**public** **class** QueueController {

//注入发送消息的对象

@Autowired

**private** JmsTemplate jmsTemplate;

//注入消息队列

@Autowired

**private** Destination destination;

//编写发送消息的方法

@RequestMapping("send/{message}")

**public** String send(@PathVariable String message) {

**this**.jmsTemplate.convertAndSend(destination, message);

**return** "消息发送成功!消息内容：" + message;

}

}

### 接收消息

编写bean，加入@Component注解让spring管理这个bean，作为接收消息的消费者

**package** cn.itcast.info.queue;

**import** org.springframework.jms.annotation.JmsListener;

**import** org.springframework.stereotype.Component;

@Component

**public** **class** Consumer {

// 接受消息方法

@JmsListener(destination = "itcast.queue")

**public** **void** readMessage(String text) {

System.***out***.println("接受到的消息是：" + text);

}

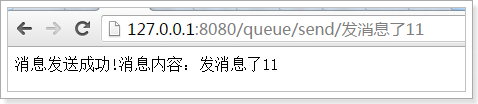
}

测试：

启动服务后，在浏览器输入以下地址

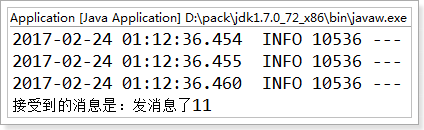
http://127.0.0.1:8080/queue/send/发消息了11

显示效果：



即可看到消息发送成功

同时可以在控制台看到以下打印信息



我们没有安装ActiveMQ，为什么可以使用？因为Spring Boot 内置了ActiveMQ 的服务，所以我们不用单独启动也可以实现消息的发送和接收。

### 使用外部服务

首先确认有一台外部ActiveMQ服务可以使用



在application.properties中加入以下配置

#ActiveMQ

spring.activemq.broker-url=tcp://192.168.37.161:61616

这样就加入了ActiveMQ服务的地址

## 整合junit

### 加入依赖

在pom.xml中加入测试依赖

<!-- 配置测试启动器 -->

<dependency>

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

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

<scope>test</scope>

</dependency>

### 编写测试类

**import** javax.jms.Destination;

**import** org.junit.Test;

**import** org.junit.runner.RunWith;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.boot.test.context.SpringBootTest;

**import** org.springframework.jms.core.JmsTemplate;

**import** org.springframework.test.context.junit4.SpringJUnit4ClassRunner;

**import** cn.itcast.info.Application;

@RunWith(SpringJUnit4ClassRunner.**class**)

@SpringBootTest(classes = Application.**class**)

**public** **class** MessageTest {

@Autowired

**private** Destination destination;

@Autowired

**private** JmsTemplate jmsTemplate;

@Test

**public** **void** test() {

System.***out***.println("我发消息了！");

**this**.jmsTemplate.convertAndSend(destination, "Hello ActiveMQ!");

}

}

SpringRunner 与SpringJUnit4ClassRunner 是继承关系，但是没有不同的地方，只是看起来子类SpringRunner要短一些而已。

@SpringBootTest 注解的class 属性要指定引导类的class

## 整合FreeMarker

### 加入依赖

<!-- FreeMarker启动器 -->

<dependency>

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

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

</dependency>

### 编写测试类

**import** java.util.Date;

**import** java.util.Map;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.web.bind.annotation.RequestMapping;

@Controller

@RequestMapping("fm")

**public** **class** FreeMarkerController {

@RequestMapping

**public** String fm(Map<String, Object> root) {

root.put("name", "张三");

root.put("date", **new** Date());

**return** "template";

}

}

### 编写模板

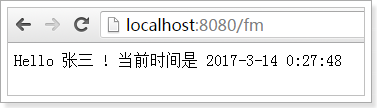
在src\main\resources\templates路径下创建template.ftl模板，内容如下：

Hello ${name} !

当前时间是

${date?datetime}

启动引导类测试效果：



## 整合Solr

### 加入依赖

<!-- solr启动器 -->

<dependency>

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

<artifactId>spring-boot-starter-data-solr</artifactId>

</dependency>

### 修改配置文件

在application.properties添加配置

#solr配置

spring.data.solr.host=http://127.0.0.1:8081/solr

### 编写Controller

**import** java.util.ArrayList;

**import** java.util.List;

**import** java.util.Map;

**import** org.apache.solr.client.solrj.SolrClient;

**import** org.apache.solr.client.solrj.SolrQuery;

**import** org.apache.solr.client.solrj.response.QueryResponse;

**import** org.apache.solr.common.SolrDocument;

**import** org.apache.solr.common.SolrDocumentList;

**import** org.apache.solr.common.SolrInputDocument;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.web.bind.annotation.PathVariable;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.ResponseBody;

@Controller

@RequestMapping("solr")

**public** **class** SolrController {

@Autowired

**private** SolrClient solrClient;

@RequestMapping("query/{data}")

**public** String query(@PathVariable String data, Map<String, Object> root) **throws** Exception {

// 查询solr

SolrQuery solrQuery = **new** SolrQuery("title:" + data);

QueryResponse response = **this**.solrClient.query(solrQuery);

SolrDocumentList results = response.getResults();

// 解析查询结果

List<String> list = **new** ArrayList<String>();

**for** (SolrDocument solrDocument : results) {

list.add(solrDocument.get("title").toString());

}

// 传递数据

root.put("total", results.getNumFound());

root.put("list", list);

**return** "solr";

}

@RequestMapping("save/{id}/{title}")

@ResponseBody

**public** String save(@PathVariable String id, @PathVariable String title) **throws** Exception {

SolrInputDocument doc = **new** SolrInputDocument();

doc.addField("id", id);

doc.addField("title", title);

**this**.solrClient.add(doc);

**this**.solrClient.commit();

**return** "保存成功";

}

}

### 编写展示模板

在src\main\resources\templates路径下创建solr.ftl模板，内容如下

查询到了${total}条数据

<br>

查询结果如下：

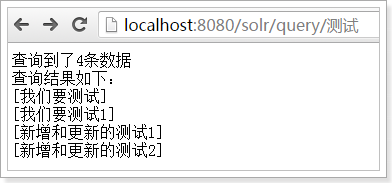
<#list list as l >

<br>${l}

</#list>

运行引导类

访问地址：http://localhost:8080/solr/query/测试，得到如下结果：



访问地址: http://localhost:8080/solr/query/save/s001/solr测试，查看索引库数据保存成功。

## 整合dubbox

### 环境准备

#### dubbox

dubbo是一个分布式的服务架构，可直接用于生产环境作为SOA服务框架。官网首页：http://dubbo.io/

淘宝将这个项目开源出来以后，得到了不少同行的支持，包括：

当当网的扩展版本dubbox ：https://github.com/dangdangdotcom/dubbox

京东的扩展版本jd-hydra: http://www.oschina.NET/p/jd-hydra

不过，略有遗憾的是， dubbo由于某些原因导致dubbo团队已经解散，已经很牛没有更新了，反到是当当网的扩展版本仍在持续发展。因为dubbox支持更新的spring版本，所以我们使用dubbox。

Dubbox在maven中央仓库并没有对应的依赖,所以我们需要自己动手将其发布到我们的本地仓库来使用。

使用git从码云上把dubbox的代码clone下来，

地址：https://git.oschina.net/wuyu15255872976/dubbox.git

执行Maven命令把工程安装到本地仓库

命令：clean install -Dmaven.test.skip

**课程资料提供的仓库已经安装好了，可以直接使用**

#### spring-boot-starter-dubbo

我们以前在使用dubbo的时候都是用的xml配置。而在整合Spring Boot的时候可以使用@ImportResource注解来引入的dubbo的xml配置。

但是Spring Boot本身并不推荐xml配置。怎么解决这个矛盾,我们可以自己准备一个Spring Boot Starter dubbo的项目来引导Spring Boot对Dubbo的自动化配置。已经有人开发好了这个自动化配置项目，我们直接使用就行了

使用git从码云上把spring-boot-starter-dubbo的代码clone下来，

地址：https://git.oschina.net/wuyu15255872976/spring-boot-starter-dubbo.git

执行Maven命令把工程安装到本地仓库

命令：clean install -Dmaven.test.skip

为了统一管理，把pom.xml修改为如下：

<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>

<parent>

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

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

<version>1.4.4.RELEASE</version>

</parent>

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

<version>1.4.4.RELEASE</version>

<name>Spring Boot Dubbo Rpc</name>

<description>Spring Boot Dubbo Rpc</description>

<url>http://projects.spring.io/spring-boot/</url>

<organization>

<name>Pivotal Software, Inc.</name>

<url>http://www.spring.io</url>

</organization>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<java.version>1.7</java.version>

</properties>

<dependencies>

<dependency>

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

<artifactId>spring-boot-actuator</artifactId>

</dependency>

<dependency>

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

<artifactId>spring-boot-configuration-processor</artifactId>

</dependency>

<dependency>

<groupId>com.alibaba</groupId>

<artifactId>dubbo</artifactId>

<version>2.8.5-SNAPSHOT</version>

<exclusions>

<exclusion>

<artifactId>spring</artifactId>

<groupId>org.springframework</groupId>

</exclusion>

</exclusions>

</dependency>

<!-- zookeeper 客户端 -->

<dependency>

<groupId>com.github.sgroschupf</groupId>

<artifactId>zkclient</artifactId>

<version>0.1</version>

</dependency>

</dependencies>

<dependencyManagement>

<dependencies>

<dependency>

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

<artifactId>spring-boot-dependencies</artifactId>

<version>1.4.4.RELEASE</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

<build>

<plugins>

<plugin>

<artifactId>maven-source-plugin</artifactId>

<configuration>

<attach>true</attach>

</configuration>

<executions>

<execution>

<phase>compile</phase>

<goals>

<goal>jar</goal>

</goals>

</execution>

</executions>

</plugin>

</plugins>

</build>

</project>

**课程资料提供的仓库已经安装好了，可以直接使用**

#### zookeeper注册中心

我们使用zookeeper作为dubbo的注册中心。

这里使用的zookeeper注册中心地址是：192.168.37.161:2181

### 搭建项目

taotao-parent作为所有工程的父工程

taotao- interface作为提供pojo和抽取服务接口的

taotao-provider作为服务提供者

taotao-consumer作为服务消费者

#### 搭建taotao-parent

创建taotao-parent，并打pom包，这里配置公共使用的依赖。

修改pom.xml为如下

<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>

<parent>

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

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

<version>1.4.4.RELEASE</version>

</parent>

<groupId>com.taotao</groupId>

<artifactId>taotao-parent</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>pom</packaging>

<properties>

<!-- 配置java版本 -->

<java.version>1.7</java.version>

</properties>

<dependencies>

<!-- 配置测试启动器 -->

<dependency>

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

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

<scope>test</scope>

</dependency>

<!-- 配置web启动器 -->

<dependency>

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

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

<optional>true</optional>

</dependency>

<!-- 配置dubbo启动器 -->

<dependency>

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

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

<version>1.4.4.RELEASE</version>

<optional>true</optional>

</dependency>

</dependencies>

</project>

#### 搭建taotao-interface

创建taotao-interface，并打jar包。

修改pom.xml为如下，几乎不用配置

<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>

<parent>

<groupId>com.taotao</groupId>

<artifactId>taotao-parent</artifactId>

<version>0.0.1-SNAPSHOT</version>

</parent>

<groupId>com.taotao</groupId>

<artifactId>taotao-interface</artifactId>

<version>0.0.1-SNAPSHOT</version>

</project>

#### 搭建taotao-provider

创建taotao-provider，并打jar包。

修改pom.xml为如下

<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>

<parent>

<groupId>com.taotao</groupId>

<artifactId>taotao-parent</artifactId>

<version>0.0.1-SNAPSHOT</version>

</parent>

<groupId>com.taotao</groupId>

<artifactId>taotao-provider</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<!-- 加入taotao-interface依赖 -->

<dependency>

<groupId>com.taotao</groupId>

<artifactId>taotao-interface</artifactId>

<version>0.0.1-SNAPSHOT</version>

</dependency>

<!-- 配置MyBatis启动器 -->

<dependency>

<groupId>org.mybatis.spring.boot</groupId>

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

<version>1.1.1</version>

</dependency>

<!-- MySQL连接驱动 -->

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

</dependency>

</dependencies>

</project>

#### 搭建taotao-consumer

创建taotao-consumer，并打jar包。

修改pom.xml为如下

<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>

<parent>

<groupId>com.taotao</groupId>

<artifactId>taotao-parent</artifactId>

<version>0.0.1-SNAPSHOT</version>

</parent>

<groupId>com.taotao</groupId>

<artifactId>taotao-consumer</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<!-- 加入taotao-interface依赖 -->

<dependency>

<groupId>com.taotao</groupId>

<artifactId>taotao-interface</artifactId>

<version>0.0.1-SNAPSHOT</version>

</dependency>

</dependencies>

</project>

## 实现功能

### 实现taotao-interface

编写pojo

**import** java.io.Serializable;

**public** **class** User **implements** Serializable {

**private** Long id;

**private** String username;

**private** String password;

**private** String name;

get/set方法

}

编写Service接口

**import** com.taotao.pojo.User;

**public** **interface** UserService {

**public** User queryUserById(Long id);

}

### 实现taotao-provider

编写UserMapper

**import** org.apache.ibatis.annotations.Mapper;

**import** org.apache.ibatis.annotations.Select;

**import** com.taotao.common.pojo.User;

@Mapper

**public** **interface** UserMapper {

@Select("select \* from user where id=#{id}")

**public** User queryUserById(Long id);

}

编写UserServiceImpl实现类

**import** org.springframework.beans.factory.annotation.Autowired;

**import** com.alibaba.dubbo.config.annotation.Service;

**import** com.taotao.common.consumer.UserService;

**import** com.taotao.common.pojo.User;

**import** com.taotao.mapper.UserMapper;

@Service

**public** **class** UserServiceImpl **implements** UserService {

@Autowired

**private** UserMapper userMapper;

@Override

**public** User queryUserById(Long id) {

User user = **this**.userMapper.queryUserById(id);

**return** user;

}

}

编写引导类

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

**import** com.alibaba.boot.dubbo.EnableDubboAutoConfiguration;

@SpringBootApplication

@EnableDubboAutoConfiguration

**public** **class** ProviderApplication {

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(ProviderApplication.**class**, args);

}

}

在src/main/resources加入配置文件application.properties

#DB Configuration:

spring.datasource.driverClassName=com.mysql.jdbc.Driver

spring.datasource.url=jdbc:mysql://127.0.0.1:3306/taotao

spring.datasource.username=root

spring.datasource.password=root

#配置服务器访问端口号

server.port=8081

#配置dubbo信息

#配置服务名称

spring.dubbo.application.name=taotao-provider

#注册中心类型

spring.dubbo.registry.protocol=zookeeper

#注册中心连接方式

spring.dubbo.registry.address=192.168.37.161:2181

#配置服务调用所使用的协议

spring.dubbo.protocol.name=dubbo

#配置服务端口号

spring.dubbo.protocol.port=20880

#配置服务访问地址

spring.dubbo.protocol.host=localhost

#配置dubbo扫描

spring.dubbo.scan=com.taotao.provider

### 实现taotao-consumer

编写Controller

**import** org.springframework.web.bind.annotation.PathVariable;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RestController;

**import** com.alibaba.dubbo.config.annotation.Reference;

**import** com.taotao.common.consumer.UserService;

**import** com.taotao.common.pojo.User;

@RestController

@RequestMapping("user")

**public** **class** UserController {

@Reference

**private** UserService userService;

@RequestMapping("{id}")

**public** User queryUserById(@PathVariable Long id) {

User user = **this**.userService.queryUserById(id);

**return** user;

}

}

编写引导类

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

**import** com.alibaba.boot.dubbo.EnableDubboAutoConfiguration;

@SpringBootApplication

@EnableDubboAutoConfiguration

**public** **class** ConsumerApplication {

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(ConsumerApplication.**class**, args);

}

}

在src/main/resources加入配置文件application.properties

#配置服务器访问端口号

server.port=8080

#配置dubbo信息

#配置服务名称

spring.dubbo.application.name=taotao-consumer

#注册中心类型

spring.dubbo.registry.protocol=zookeeper

#注册中心连接方式

spring.dubbo.registry.address=192.168.37.161:2181

#配置服务调用所使用的协议

spring.dubbo.protocol.name=dubbo

#配置服务端口号

spring.dubbo.protocol.port=20880

#配置服务访问地址

spring.dubbo.protocol.host=localhost

#配置dubbo扫描

spring.dubbo.scan=com.taotao.consumer

# Spring Boot 部署

## 打jar包

在工程的pom.xml中添加以下依赖

<build>

<plugins>

<plugin>

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

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

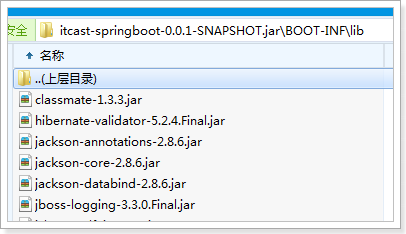
</plugins>

</build>

我们目前的工程采用的是jar 的打包方式，所以我们在执行package 命令后，

会产生一个jar 包。

我们进入到这个目录用压缩软件打开此jar 包，其中我们发现了一个叫lib 的文件夹，打开lib 文件夹发现此文件夹下全是工程依赖的jar 包，甚至还有tomcat。这种包含有jar 包的jar包，我们称之为fatJAR( 胖jar 包)



由于fatJAR 本身就包括tomcat , 我们就不需要另外部署了，直接在命令行就可以把我们的应用启动起来，在命令行，进入到jar 包所在的目录，我们可以通过以下java –jar命令来执行此jar 包。

在控制台会出现启动信息，在浏览器访问程序

## 打war包

spring-boot 默认提供内嵌的tomcat，所以打包直接生成jar 包，用java

-jar 命令就可以启动。但是，有时候我们更希望一个tomcat 来管理多个项目，

这种情况下就需要项目是war 格式的包而不是jar 格式的包。

我们按照以下步骤完成对工程的改造

（1）修改pom.xml

将打包方式修改为war

<packaging>war</packaging>

添加依赖

<dependency>

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

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

<scope>provided</scope>

</dependency>

spring-boot-starter-tomcat 是原来被传递过来的依赖，默认会打到包里，所

以我们再次引入此依赖，并指定依赖范围为provided，这样tomcat 相关的jar

就不会打包到war 里了.

（2）添加ServletInitializer

import org.springframework.boot.builder.SpringApplicationBuilder;

import

org.springframework.boot.context.web.SpringBootServletInitializer;

public class ServletInitializer extends SpringBootServletInitializer {

@Override

protected SpringApplicationBuilder configure(SpringApplicationBuilder

application) {

return application.sources(Application.class);

}

}

由于我们采用web3.0 规范，是没有web.xml 的，而此类的作用与web.xml

相同。

（3）运行package 打包命令生成war 包

生成后将war 包放入tomcat，启动tomcat，测试完成的功能是否可以使用。

# 附录1. Spring-boot的起步依赖

(摘自Spring-boot 1.4.4官方文档)

1. **Spring Boot application starters**

spring-boot-starter-thymeleaf

使用Thymeleaf视图构建MVC Web应用程序

spring-boot-starter-ws

使用Spring Web服务。1.4不推荐使用，推荐使用spring-boot-starter-web-services

spring-boot-starter-data-couchbase

Starter for using Couchbase document-oriented database and Spring Data Couchbase

spring-boot-starter-artemis

使用Apache Artemis启动JMS消息传递

spring-boot-starter-web-services

使用Spring Web服务

spring-boot-starter-mail

支持使用Java Mail和Spring Framework发送电子邮件

spring-boot-starter-data-redis

使用Redis键值数据存储与Spring Data Redis和Jedis客户端

spring-boot-starter-web

启动器构建web，包括RESTful，使用Spring MVC的应用程序。使用Tomcat作为默认嵌入式容器

spring-boot-starter-data-gemfire

Starter for using GemFire distributed data store and Spring Data GemFire

spring-boot-starter-activemq

使用Apache ActiveMQ启动JMS消息传递

spring-boot-starter-data-elasticsearch

使用Elasticsearch搜索和分析引擎和Spring Data Elasticsearch

spring-boot-starter-integration

Starter for using Spring Integration

spring-boot-starter-test

Spring Boot应用程序用于测试包括JUnit，Hamcrest和Mockito

spring-boot-starter-hornetq

使用HornetQ启动JMS消息传递。1.4已弃用，推荐使用spring-boot-starter-artemis

spring-boot-starter-jdbc

使用JDBC与Tomcat JDBC连接池

spring-boot-starter-mobile

使用Spring Mobile构建Web应用程序的入门

spring-boot-starter-validation

使用Java Bean校验与Hibernate校验器

spring-boot-starter-hateoas

使用Spring MVC和Spring HATEOAS构建基于超媒体的RESTful Web应用程序的入门

spring-boot-starter-jersey

使用JAX-RS和Jersey构建RESTful Web应用程序的入门。 spring-boot-starter-web的替代品

spring-boot-starter-data-neo4j

使用Neo4j图数据库和Spring Data Neo4j

spring-boot-starter-websocket

使用Spring Framework的WebSocket支持构建WebSocket应用程序

spring-boot-starter-aop

使用Spring AOP和AspectJ进行面向方面编程

spring-boot-starter-amqp

使用Spring AMQP和Rabbit MQ的入门

spring-boot-starter-data-cassandra

使用Cassandra分布式数据库和Spring Data Cassandra

spring-boot-starter-social-facebook

使用Spring Social Facebook

spring-boot-starter-jta-atomikos

使用Atomikos进行JTA事务

spring-boot-starter-security

使用Spring Security

spring-boot-starter-mustache

使用Mustache视图构建MVC Web应用程序

spring-boot-starter-data-jpa

使用Spring Data JPA与Hibernate

spring-boot-starter

核心启动器，包括自动配置支持，日志记录和YAML

spring-boot-starter-velocity

使用Velocity视图构建MVC Web应用程序。1.4已弃用

spring-boot-starter-groovy-templates

使用Groovy模板视图构建MVC Web应用程序

spring-boot-starter-freemarker

使用FreeMarker视图构建MVC Web应用程序

spring-boot-starter-batch

使用Spring Batch

spring-boot-starter-redis

使用Redis键值数据存储与Spring Data Redis和Jedis客户端的入门。1.4已弃用，建议使用spring-boot-starter-data-redis

spring-boot-starter-social-linkedin

Stater for using Spring Social LinkedIn

spring-boot-starter-cache

支持使用Spring Framework的缓存

spring-boot-starter-data-solr

使用带有Spring Data Solr的Apache Solr搜索平台

spring-boot-starter-data-mongodb

使用MongoDB和Spring Data MongoDB

spring-boot-starter-jooq

使用jOOQ访问SQL数据库。 spring-boot-starter-data-jpa或spring-boot-starter-jdbc的替代方法

spring-boot-starter-jta-narayana

Spring Boot启动Narayana JTA

spring-boot-starter-cloud-connectors

启动者使用Spring Cloud连接器，简化了连接到云平台中的服务，如Cloud Foundry和Heroku

spring-boot-starter-jta-bitronix

使用Bitronix进行JTA事务

spring-boot-starter-social-twitter

使用Spring Social Twitter

spring-boot-starter-data-rest

使用Spring Data REST通过REST暴露Spring数据存储库

1. **Spring Boot production starters**

spring-boot-starter-actuator

使用Spring Boot的Actuator，提供生产就绪的功能，以帮助您监视和管理您的应用程序

spring-boot-starter-remote-shell

使用CRaSH远程shell通过SSH监视和管理您的应用程序

1. **Spring Boot technical starters**

spring-boot-starter-undertow

使用Undertow作为嵌入式servlet容器。 spring-boot-starter-tomcat的替代方法

spring-boot-starter-jetty

使用Jetty作为嵌入式servlet容器的。 spring-boot-starter-tomcat的替代方法

spring-boot-starter-logging

使用Logback进行日志记录。 默认日志启动器

spring-boot-starter-tomcat

使用Tomcat作为嵌入式servlet容器。 spring-boot-starter-web使用的默认servlet容器

spring-boot-starter-log4j2

使用Log4j2进行日志记录。 spring-boot-starter-logging的替代方法

# 附录2. Spring-boot的application.properties

(摘自Spring-boot 1.4.4官方文档)

*# ===================================================================*

*# COMMON SPRING BOOT PROPERTIES*

*#*

*# This sample file is provided as a guideline. Do NOT copy it in its*

*# entirety to your own application. ^^^*

*# ===================================================================*

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

*# CORE PROPERTIES*

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

*# BANNER*

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

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

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

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

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

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

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

*# LOGGING*

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

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

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

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

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

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

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

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

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

*# AOP*

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

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

*# IDENTITY (*[ContextIdApplicationContextInitializer](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot/src/main/java/org/springframework/boot/context/ContextIdApplicationContextInitializer.java))

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

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

*# ADMIN (*[SpringApplicationAdminJmxAutoConfiguration](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/admin/SpringApplicationAdminJmxAutoConfiguration.java))

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

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

*# AUTO-CONFIGURATION*

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

*# SPRING CORE*

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

*# SPRING CACHE (*[CacheProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/cache/CacheProperties.java))

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

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

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

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

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

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

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

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

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

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

*# SPRING CONFIG - using environment property only (*[ConfigFileApplicationListener](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot/src/main/java/org/springframework/boot/context/config/ConfigFileApplicationListener.java))

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

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

*# HAZELCAST (*[HazelcastProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/hazelcast/HazelcastProperties.java))

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

*# PROJECT INFORMATION (*[ProjectInfoProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/info/ProjectInfoProperties.java))

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

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

*# JMX*

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

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

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

*# Email (*[MailProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/mail/MailProperties.java))

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

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

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

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

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

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

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

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

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

*# APPLICATION SETTINGS (*[SpringApplication](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot/src/main/java/org/springframework/boot/SpringApplication.java))

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

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

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

*# FILE ENCODING (*[FileEncodingApplicationListener](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot/src/main/java/org/springframework/boot/context/FileEncodingApplicationListener.java))

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

*# INTERNATIONALIZATION (*[MessageSourceAutoConfiguration](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/MessageSourceAutoConfiguration.java))

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

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

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

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

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

*# OUTPUT*

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

*# PID FILE (*[ApplicationPidFileWriter](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-actuator/src/main/java/org/springframework/boot/actuate/system/ApplicationPidFileWriter.java))

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

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

*# PROFILES*

spring.profiles.active= *# Comma-separated list of* [active profiles](http://docs.spring.io/spring-boot/docs/1.4.4.RELEASE/reference/htmlsingle/#howto-set-active-spring-profiles).

spring.profiles.include= *# Unconditionally activate the specified comma separated profiles.*

*# SENDGRID (*[SendGridAutoConfiguration](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/sendgrid/SendGridAutoConfiguration.java))

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

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

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

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

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

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

*# WEB PROPERTIES*

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

*# EMBEDDED SERVER CONFIGURATION (*[ServerProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/web/ServerProperties.java))

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

*# FREEMARKER (*[FreeMarkerAutoConfiguration](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/freemarker/FreeMarkerAutoConfiguration.java))

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

*# GROOVY TEMPLATES (*[GroovyTemplateAutoConfiguration](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/groovy/template/GroovyTemplateAutoConfiguration.java))

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

*# SPRING HATEOAS (*[HateoasProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/hateoas/HateoasProperties.java))

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

*# HTTP message conversion*

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

*# HTTP encoding (*[HttpEncodingProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/web/HttpEncodingProperties.java))

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

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

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

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

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

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

*# MULTIPART (*[MultipartProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/web/MultipartProperties.java))

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

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

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

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

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

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

*# JACKSON (*[JacksonProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/jackson/JacksonProperties.java))

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

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

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

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

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

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

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

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

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

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

spring.jackson.serialization-inclusion= *# Controls the inclusion of properties during serialization. Configured with one of the values in Jackson's JsonInclude.Include enumeration.*

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

*# JERSEY (*[JerseyProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/jersey/JerseyProperties.java))

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

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

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

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

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

*# SPRING MOBILE DEVICE VIEWS (*[DeviceDelegatingViewResolverAutoConfiguration](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/mobile/DeviceDelegatingViewResolverAutoConfiguration.java))

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

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

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

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

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

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

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

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

*# SPRING MOBILE SITE PREFERENCE (*[SitePreferenceAutoConfiguration](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/mobile/SitePreferenceAutoConfiguration.java))

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

*# MUSTACHE TEMPLATES (*[MustacheAutoConfiguration](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/mustache/MustacheAutoConfiguration.java))

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

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

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

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

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

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

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

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

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

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

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

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

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

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

*# SPRING MVC (*[WebMvcProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/web/WebMvcProperties.java))

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

*# SPRING RESOURCES HANDLING (*[ResourceProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/web/ResourceProperties.java))

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

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

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

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

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

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

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

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

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

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

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

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

*# SPRING SESSION (*[SessionProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/session/SessionProperties.java))

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

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

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

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

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

spring.session.redis.flush-mode= *# Flush mode for the Redis sessions.*

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

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

*# SPRING SOCIAL (*[SocialWebAutoConfiguration](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/social/SocialWebAutoConfiguration.java))

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

*# SPRING SOCIAL FACEBOOK (*[FacebookAutoConfiguration](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/social/FacebookAutoConfiguration.java))

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

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

*# SPRING SOCIAL LINKEDIN (*[LinkedInAutoConfiguration](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/social/LinkedInAutoConfiguration.java))

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

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

*# SPRING SOCIAL TWITTER (*[TwitterAutoConfiguration](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/social/TwitterAutoConfiguration.java))

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

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

*# THYMELEAF (*[ThymeleafAutoConfiguration](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/thymeleaf/ThymeleafAutoConfiguration.java))

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

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

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

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

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

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

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

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

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

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

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

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

*# VELOCITY TEMPLATES (*[VelocityAutoConfiguration](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/velocity/VelocityAutoConfiguration.java))

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

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

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

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

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

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

spring.velocity.date-tool-attribute= *# Name of the DateTool helper object to expose in the Velocity context of the view.*

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

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

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

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

spring.velocity.number-tool-attribute= *# Name of the NumberTool helper object to expose in the Velocity context of the view.*

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

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

spring.velocity.properties.\*= *# Additional velocity properties.*

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

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

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

spring.velocity.toolbox-config-location= *# Velocity Toolbox config location. For instance `/WEB-INF/toolbox.xml`*

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

*# SPRING WEB SERVICES (*[WebServicesProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/webservices/WebServicesProperties.java))

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

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

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

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

*# SECURITY PROPERTIES*

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

*# SECURITY (*[SecurityProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/security/SecurityProperties.java))

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

*# SECURITY OAUTH2 CLIENT (*[OAuth2ClientProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/security/oauth2/OAuth2ClientProperties.java))

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

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

*# SECURITY OAUTH2 RESOURCES (*[ResourceServerProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/security/oauth2/resource/ResourceServerProperties.java))

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

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

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

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

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

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

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

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

*# SECURITY OAUTH2 SSO (*[OAuth2SsoProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/security/oauth2/client/OAuth2SsoProperties.java))

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

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

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

*# DATA PROPERTIES*

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

*# FLYWAY (*[FlywayProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/flyway/FlywayProperties.java))

flyway.baseline-description= *#*

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

flyway.baseline-on-migrate= *#*

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

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

flyway.enabled=true *# Enable flyway.*

flyway.encoding= *#*

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

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

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

flyway.out-of-order= *#*

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

flyway.placeholder-prefix= *#*

flyway.placeholder-replacement= *#*

flyway.placeholder-suffix= *#*

flyway.placeholders.\*= *#*

flyway.schemas= *# schemas to update*

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

flyway.sql-migration-separator= *#*

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

flyway.table= *#*

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

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

flyway.validate-on-migrate= *#*

*# LIQUIBASE (*[LiquibaseProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/liquibase/LiquibaseProperties.java))

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

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

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

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

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

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

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

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

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

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

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

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

*# COUCHBASE (*[CouchbaseProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/couchbase/CouchbaseProperties.java))

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

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

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

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

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

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

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

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

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

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

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

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

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

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

*# DAO (*[PersistenceExceptionTranslationAutoConfiguration](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/dao/PersistenceExceptionTranslationAutoConfiguration.java))

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

*# CASSANDRA (*[CassandraProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/cassandra/CassandraProperties.java))

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

spring.data.cassandra.compression= *# Compression supported by the Cassandra binary protocol.*

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

*# DATA COUCHBASE (*[CouchbaseDataProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/data/couchbase/CouchbaseDataProperties.java))

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

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

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

*# ELASTICSEARCH (*[ElasticsearchProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/data/elasticsearch/ElasticsearchProperties.java))

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

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

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

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

*# MONGODB (*[MongoProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/mongo/MongoProperties.java))

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

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

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

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

spring.data.mongodb.host=localhost *# Mongo server host.*

spring.data.mongodb.password= *# Login password of the mongo server.*

spring.data.mongodb.port=27017 *# Mongo server port.*

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

spring.data.mongodb.uri=mongodb://localhost/test *# Mongo database URI. When set, host and port are ignored.*

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

*# DATA REDIS*

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

*# NEO4J (*[Neo4jProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/neo4j/Neo4jProperties.java))

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

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

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

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

spring.data.neo4j.session.scope=singleton *# Scope (lifetime) of the session.*

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

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

*# DATA REST (*[RepositoryRestProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/data/rest/RepositoryRestProperties.java))

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

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

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

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

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

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

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

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

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

*# SOLR (*[SolrProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/solr/SolrProperties.java))

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

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

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

*# DATASOURCE (*[DataSourceAutoConfiguration](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/jdbc/DataSourceAutoConfiguration.java) & [DataSourceProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/jdbc/DataSourceProperties.java))

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

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

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

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

spring.datasource.dbcp.\*= *# Commons DBCP specific settings*

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

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

spring.datasource.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 reference.*

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

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

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

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

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

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

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

spring.datasource.username=

*# JEST (Elasticsearch HTTP client) (*[JestProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/jest/JestProperties.java))

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

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

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

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

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

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

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

*# H2 Web Console (*[H2ConsoleProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/h2/H2ConsoleProperties.java))

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

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

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

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

*# JOOQ (*[JooqAutoConfiguration](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/jooq/JooqAutoConfiguration.java))

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

*# JPA (*[JpaBaseConfiguration](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/orm/jpa/JpaBaseConfiguration.java), [HibernateJpaAutoConfiguration](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/orm/jpa/HibernateJpaAutoConfiguration.java))

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

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

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

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

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

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

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

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

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

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

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

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

*# JTA (*[JtaAutoConfiguration](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/transaction/jta/JtaAutoConfiguration.java))

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

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

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

*# ATOMIKOS (*[AtomikosProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot/src/main/java/org/springframework/boot/jta/atomikos/AtomikosProperties.java))

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

*# BITRONIX*

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

*# NARAYANA (*[NarayanaProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot/src/main/java/org/springframework/boot/jta/narayana/NarayanaProperties.java))

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

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

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

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

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

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

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

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

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

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

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

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

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

*# EMBEDDED MONGODB (*[EmbeddedMongoProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/mongo/embedded/EmbeddedMongoProperties.java))

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

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

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

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

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

*# REDIS (*[RedisProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/data/redis/RedisProperties.java))

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

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

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

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

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

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

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

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

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

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

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

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

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

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

*# INTEGRATION PROPERTIES*

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

*# ACTIVEMQ (*[ActiveMQProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/jms/activemq/ActiveMQProperties.java))

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

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

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

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

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

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

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

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

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

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

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

*# ARTEMIS (*[ArtemisProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/jms/artemis/ArtemisProperties.java))

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

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

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

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

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

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

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

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

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

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

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

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

*# SPRING BATCH (*[BatchProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/batch/BatchProperties.java))

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

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

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

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

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

*# HORNETQ (*[HornetQProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/jms/hornetq/HornetQProperties.java))

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

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

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

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

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

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

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

spring.hornetq.host=localhost *# HornetQ broker host.*

spring.hornetq.mode= *# HornetQ deployment mode, auto-detected by default.*

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

spring.hornetq.port=5445 *# HornetQ broker port.*

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

*# JMS (*[JmsProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/jms/JmsProperties.java))

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

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

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

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

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

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

*# RABBIT (*[RabbitProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/amqp/RabbitProperties.java))

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

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

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

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

spring.rabbitmq.cache.connection.size= *# Number of connections to cache.*

spring.rabbitmq.connection-timeout= *# Connection timeout, in milliseconds; zero for infinite.*

spring.rabbitmq.dynamic=true *# Create an AmqpAdmin bean.*

spring.rabbitmq.host=localhost *# RabbitMQ host.*

spring.rabbitmq.listener.acknowledge-mode= *# Acknowledge mode of container.*

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

spring.rabbitmq.listener.concurrency= *# Minimum number of consumers.*

spring.rabbitmq.listener.default-requeue-rejected= *# Whether or not to requeue delivery failures; default `true`.*

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

spring.rabbitmq.listener.prefetch= *# Number of messages to be handled in a single request. It should be greater than or equal to the transaction size (if used).*

spring.rabbitmq.listener.retry.enabled=false *# Whether or not publishing retries are enabled.*

spring.rabbitmq.listener.retry.initial-interval=1000 *# Interval between the first and second attempt to deliver a message.*

spring.rabbitmq.listener.retry.max-attempts=3 *# Maximum number of attempts to deliver a message.*

spring.rabbitmq.listener.retry.max-interval=10000 *# Maximum interval between attempts.*

spring.rabbitmq.listener.retry.multiplier=1.0 *# A multiplier to apply to the previous delivery retry interval.*

spring.rabbitmq.listener.retry.stateless=true *# Whether or not retry is stateless or stateful.*

spring.rabbitmq.listener.transaction-size= *# Number of messages to be processed in a transaction. For best results it should be less than or equal to the prefetch count.*

spring.rabbitmq.password= *# Login to authenticate against the broker.*

spring.rabbitmq.port=5672 *# RabbitMQ port.*

spring.rabbitmq.publisher-confirms=false *# Enable publisher confirms.*

spring.rabbitmq.publisher-returns=false *# Enable publisher returns.*

spring.rabbitmq.requested-heartbeat= *# Requested heartbeat timeout, in seconds; zero for none.*

spring.rabbitmq.ssl.enabled=false *# Enable SSL support.*

spring.rabbitmq.ssl.key-store= *# Path to the key store that holds the SSL certificate.*

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

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

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

spring.rabbitmq.ssl.algorithm= *# SSL algorithm to use. By default configure by the rabbit client library.*

spring.rabbitmq.template.mandatory=false *# Enable mandatory messages.*

spring.rabbitmq.template.receive-timeout=0 *# Timeout for `receive()` methods.*

spring.rabbitmq.template.reply-timeout=5000 *# Timeout for `sendAndReceive()` methods.*

spring.rabbitmq.template.retry.enabled=false *# Set to true to enable retries in the `RabbitTemplate`.*

spring.rabbitmq.template.retry.initial-interval=1000 *# Interval between the first and second attempt to publish a message.*

spring.rabbitmq.template.retry.max-attempts=3 *# Maximum number of attempts to publish a message.*

spring.rabbitmq.template.retry.max-interval=10000 *# Maximum number of attempts to publish a message.*

spring.rabbitmq.template.retry.multiplier=1.0 *# A multiplier to apply to the previous publishing retry interval.*

spring.rabbitmq.username= *# Login user to authenticate to the broker.*

spring.rabbitmq.virtual-host= *# Virtual host to use when connecting to the broker.*

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

*# ACTUATOR PROPERTIES*

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

*# ENDPOINTS (*[AbstractEndpoint](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-actuator/src/main/java/org/springframework/boot/actuate/endpoint/AbstractEndpoint.java) subclasses)

endpoints.enabled=true *# Enable endpoints.*

endpoints.sensitive= *# Default endpoint sensitive setting.*

endpoints.actuator.enabled=true *# Enable the endpoint.*

endpoints.actuator.path= *# Endpoint URL path.*

endpoints.actuator.sensitive=false *# Enable security on the endpoint.*

endpoints.autoconfig.enabled= *# Enable the endpoint.*

endpoints.autoconfig.id= *# Endpoint identifier.*

endpoints.autoconfig.path= *# Endpoint path.*

endpoints.autoconfig.sensitive= *# Mark if the endpoint exposes sensitive information.*

endpoints.beans.enabled= *# Enable the endpoint.*

endpoints.beans.id= *# Endpoint identifier.*

endpoints.beans.path= *# Endpoint path.*

endpoints.beans.sensitive= *# Mark if the endpoint exposes sensitive information.*

endpoints.configprops.enabled= *# Enable the endpoint.*

endpoints.configprops.id= *# Endpoint identifier.*

endpoints.configprops.keys-to-sanitize=password,secret,key,token,.\*credentials.\*,vcap\_services *# Keys that should be sanitized. Keys can be simple strings that the property ends with or regex expressions.*

endpoints.configprops.path= *# Endpoint path.*

endpoints.configprops.sensitive= *# Mark if the endpoint exposes sensitive information.*

endpoints.docs.curies.enabled=false *# Enable the curie generation.*

endpoints.docs.enabled=true *# Enable actuator docs endpoint.*

endpoints.docs.path=/docs *#*

endpoints.docs.sensitive=false *#*

endpoints.dump.enabled= *# Enable the endpoint.*

endpoints.dump.id= *# Endpoint identifier.*

endpoints.dump.path= *# Endpoint path.*

endpoints.dump.sensitive= *# Mark if the endpoint exposes sensitive information.*

endpoints.env.enabled= *# Enable the endpoint.*

endpoints.env.id= *# Endpoint identifier.*

endpoints.env.keys-to-sanitize=password,secret,key,token,.\*credentials.\*,vcap\_services *# Keys that should be sanitized. Keys can be simple strings that the property ends with or regex expressions.*

endpoints.env.path= *# Endpoint path.*

endpoints.env.sensitive= *# Mark if the endpoint exposes sensitive information.*

endpoints.flyway.enabled= *# Enable the endpoint.*

endpoints.flyway.id= *# Endpoint identifier.*

endpoints.flyway.sensitive= *# Mark if the endpoint exposes sensitive information.*

endpoints.health.enabled= *# Enable the endpoint.*

endpoints.health.id= *# Endpoint identifier.*

endpoints.health.mapping.\*= *# Mapping of health statuses to HttpStatus codes. By default, registered health statuses map to sensible defaults (i.e. UP maps to 200).*

endpoints.health.path= *# Endpoint path.*

endpoints.health.sensitive= *# Mark if the endpoint exposes sensitive information.*

endpoints.health.time-to-live=1000 *# Time to live for cached result, in milliseconds.*

endpoints.heapdump.enabled= *# Enable the endpoint.*

endpoints.heapdump.path= *# Endpoint path.*

endpoints.heapdump.sensitive= *# Mark if the endpoint exposes sensitive information.*

endpoints.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.mappings.enabled= *# Enable the endpoint.*

endpoints.mappings.id= *# Endpoint identifier.*

endpoints.mappings.path= *# Endpoint path.*

endpoints.mappings.sensitive= *# Mark if the endpoint exposes sensitive information.*

endpoints.metrics.enabled= *# Enable the endpoint.*

endpoints.metrics.filter.enabled=true *# Enable the metrics servlet filter.*

endpoints.metrics.filter.gauge-submissions=merged *# Http filter gauge submissions (merged, per-http-method)*

endpoints.metrics.filter.counter-submissions=merged *# Http filter counter submissions (merged, per-http-method)*

endpoints.metrics.id= *# Endpoint identifier.*

endpoints.metrics.path= *# Endpoint path.*

endpoints.metrics.sensitive= *# Mark if the endpoint exposes sensitive information.*

endpoints.shutdown.enabled= *# Enable the endpoint.*

endpoints.shutdown.id= *# Endpoint identifier.*

endpoints.shutdown.path= *# Endpoint path.*

endpoints.shutdown.sensitive= *# Mark if the endpoint exposes sensitive information.*

endpoints.trace.enabled= *# Enable the endpoint.*

endpoints.trace.id= *# Endpoint identifier.*

endpoints.trace.path= *# Endpoint path.*

endpoints.trace.sensitive= *# Mark if the endpoint exposes sensitive information.*

*# ENDPOINTS CORS CONFIGURATION (*[EndpointCorsProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-actuator/src/main/java/org/springframework/boot/actuate/autoconfigure/EndpointCorsProperties.java))

endpoints.cors.allow-credentials= *# Set whether credentials are supported. When not set, credentials are not supported.*

endpoints.cors.allowed-headers= *# Comma-separated list of headers to allow in a request. '\*' allows all headers.*

endpoints.cors.allowed-methods=GET *# Comma-separated list of methods to allow. '\*' allows all methods.*

endpoints.cors.allowed-origins= *# Comma-separated list of origins to allow. '\*' allows all origins. When not set, CORS support is disabled.*

endpoints.cors.exposed-headers= *# Comma-separated list of headers to include in a response.*

endpoints.cors.max-age=1800 *# How long, in seconds, the response from a pre-flight request can be cached by clients.*

*# JMX ENDPOINT (*[EndpointMBeanExportProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-actuator/src/main/java/org/springframework/boot/actuate/autoconfigure/EndpointMBeanExportProperties.java))

endpoints.jmx.domain= *# JMX domain name. Initialized with the value of 'spring.jmx.default-domain' if set.*

endpoints.jmx.enabled=true *# Enable JMX export of all endpoints.*

endpoints.jmx.static-names= *# Additional static properties to append to all ObjectNames of MBeans representing Endpoints.*

endpoints.jmx.unique-names=false *# Ensure that ObjectNames are modified in case of conflict.*

*# JOLOKIA (*[JolokiaProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-actuator/src/main/java/org/springframework/boot/actuate/autoconfigure/JolokiaProperties.java))

jolokia.config.\*= *# See Jolokia manual*

*# MANAGEMENT HTTP SERVER (*[ManagementServerProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-actuator/src/main/java/org/springframework/boot/actuate/autoconfigure/ManagementServerProperties.java))

management.add-application-context-header=true *# Add the "X-Application-Context" HTTP header in each response.*

management.address= *# Network address that the management endpoints should bind to.*

management.context-path= *# Management endpoint context-path. For instance `/actuator`*

management.port= *# Management endpoint HTTP port. Uses the same port as the application by default. Configure a different port to use management-specific SSL.*

management.security.enabled=true *# Enable security.*

management.security.roles=ADMIN *# Comma-separated list of roles that can access the management endpoint.*

management.security.sessions=stateless *# Session creating policy to use (always, never, if\_required, stateless).*

management.ssl.ciphers= *# Supported SSL ciphers. Requires a custom management.port.*

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

management.ssl.enabled= *# Enable SSL support. Requires a custom management.port.*

management.ssl.enabled-protocols= *# Enabled SSL protocols. Requires a custom management.port.*

management.ssl.key-alias= *# Alias that identifies the key in the key store. Requires a custom management.port.*

management.ssl.key-password= *# Password used to access the key in the key store. Requires a custom management.port.*

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

management.ssl.key-store-password= *# Password used to access the key store. Requires a custom management.port.*

management.ssl.key-store-provider= *# Provider for the key store. Requires a custom management.port.*

management.ssl.key-store-type= *# Type of the key store. Requires a custom management.port.*

management.ssl.protocol=TLS *# SSL protocol to use. Requires a custom management.port.*

management.ssl.trust-store= *# Trust store that holds SSL certificates. Requires a custom management.port.*

management.ssl.trust-store-password= *# Password used to access the trust store. Requires a custom management.port.*

management.ssl.trust-store-provider= *# Provider for the trust store. Requires a custom management.port.*

management.ssl.trust-store-type= *# Type of the trust store. Requires a custom management.port.*

*# HEALTH INDICATORS (previously health.\*)*

management.health.db.enabled=true *# Enable database health check.*

management.health.defaults.enabled=true *# Enable default health indicators.*

management.health.diskspace.enabled=true *# Enable disk space health check.*

management.health.diskspace.path= *# Path used to compute the available disk space.*

management.health.diskspace.threshold=0 *# Minimum disk space that should be available, in bytes.*

management.health.elasticsearch.enabled=true *# Enable elasticsearch health check.*

management.health.elasticsearch.indices= *# Comma-separated index names.*

management.health.elasticsearch.response-timeout=100 *# The time, in milliseconds, to wait for a response from the cluster.*

management.health.jms.enabled=true *# Enable JMS health check.*

management.health.mail.enabled=true *# Enable Mail health check.*

management.health.mongo.enabled=true *# Enable MongoDB health check.*

management.health.rabbit.enabled=true *# Enable RabbitMQ health check.*

management.health.redis.enabled=true *# Enable Redis health check.*

management.health.solr.enabled=true *# Enable Solr health check.*

management.health.status.order=DOWN, OUT\_OF\_SERVICE, UP, UNKNOWN *# Comma-separated list of health statuses in order of severity.*

*# INFO CONTRIBUTORS (*[InfoContributorProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-actuator/src/main/java/org/springframework/boot/actuate/autoconfigure/InfoContributorProperties.java))

management.info.build.enabled=true *# Enable build info.*

management.info.defaults.enabled=true *# Enable default info contributors.*

management.info.env.enabled=true *# Enable environment info.*

management.info.git.enabled=true *# Enable git info.*

management.info.git.mode=simple *# Mode to use to expose git information.*

*# REMOTE SHELL (*[ShellProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-actuator/src/main/java/org/springframework/boot/actuate/autoconfigure/ShellProperties.java))

management.shell.auth.type=simple *# Authentication type. Auto-detected according to the environment.*

management.shell.auth.jaas.domain=my-domain *# JAAS domain.*

management.shell.auth.key.path= *# Path to the authentication key. This should point to a valid ".pem" file.*

management.shell.auth.simple.user.name=user *# Login user.*

management.shell.auth.simple.user.password= *# Login password.*

management.shell.auth.spring.roles=ADMIN *# Comma-separated list of required roles to login to the CRaSH console.*

management.shell.command-path-patterns=classpath\*:/commands/\*\*,classpath\*:/crash/commands/\*\* *# Patterns to use to look for commands.*

management.shell.command-refresh-interval=-1 *# Scan for changes and update the command if necessary (in seconds).*

management.shell.config-path-patterns=classpath\*:/crash/\* *# Patterns to use to look for configurations.*

management.shell.disabled-commands=jpa\*,jdbc\*,jndi\* *# Comma-separated list of commands to disable.*

management.shell.disabled-plugins= *# Comma-separated list of plugins to disable. Certain plugins are disabled by default based on the environment.*

management.shell.ssh.auth-timeout = *# Number of milliseconds after user will be prompted to login again.*

management.shell.ssh.enabled=true *# Enable CRaSH SSH support.*

management.shell.ssh.idle-timeout = *# Number of milliseconds after which unused connections are closed.*

management.shell.ssh.key-path= *# Path to the SSH server key.*

management.shell.ssh.port=2000 *# SSH port.*

management.shell.telnet.enabled=false *# Enable CRaSH telnet support. Enabled by default if the TelnetPlugin is available.*

management.shell.telnet.port=5000 *# Telnet port.*

*# TRACING (*[TraceProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-actuator/src/main/java/org/springframework/boot/actuate/trace/TraceProperties.java))

management.trace.include=request-headers,response-headers,cookies,errors *# Items to be included in the trace.*

*# METRICS EXPORT (*[MetricExportProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-actuator/src/main/java/org/springframework/boot/actuate/metrics/export/MetricExportProperties.java))

spring.metrics.export.aggregate.key-pattern= *# Pattern that tells the aggregator what to do with the keys from the source repository.*

spring.metrics.export.aggregate.prefix= *# Prefix for global repository if active.*

spring.metrics.export.delay-millis=5000 *# Delay in milliseconds between export ticks. Metrics are exported to external sources on a schedule with this delay.*

spring.metrics.export.enabled=true *# Flag to enable metric export (assuming a MetricWriter is available).*

spring.metrics.export.excludes= *# List of patterns for metric names to exclude. Applied after the includes.*

spring.metrics.export.includes= *# List of patterns for metric names to include.*

spring.metrics.export.redis.key=keys.spring.metrics *# Key for redis repository export (if active).*

spring.metrics.export.redis.prefix=spring.metrics *# Prefix for redis repository if active.*

spring.metrics.export.send-latest= *# Flag to switch off any available optimizations based on not exporting unchanged metric values.*

spring.metrics.export.statsd.host= *# Host of a statsd server to receive exported metrics.*

spring.metrics.export.statsd.port=8125 *# Port of a statsd server to receive exported metrics.*

spring.metrics.export.statsd.prefix= *# Prefix for statsd exported metrics.*

spring.metrics.export.triggers.\*= *# Specific trigger properties per MetricWriter bean name.*

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

*# DEVTOOLS PROPERTIES*

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

*# DEVTOOLS (*[DevToolsProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-devtools/src/main/java/org/springframework/boot/devtools/autoconfigure/DevToolsProperties.java))

spring.devtools.livereload.enabled=true *# Enable a livereload.com compatible server.*

spring.devtools.livereload.port=35729 *# Server port.*

spring.devtools.restart.additional-exclude= *# Additional patterns that should be excluded from triggering a full restart.*

spring.devtools.restart.additional-paths= *# Additional paths to watch for changes.*

spring.devtools.restart.enabled=true *# Enable automatic restart.*

spring.devtools.restart.exclude=META-INF/maven/\*\*,META-INF/resources/\*\*,resources/\*\*,static/\*\*,public/\*\*,templates/\*\*,\*\*/\*Test.class,\*\*/\*Tests.class,git.properties *# Patterns that should be excluded from triggering a full restart.*

spring.devtools.restart.poll-interval=1000 *# Amount of time (in milliseconds) to wait between polling for classpath changes.*

spring.devtools.restart.quiet-period=400 *# Amount of quiet time (in milliseconds) required without any classpath changes before a restart is triggered.*

spring.devtools.restart.trigger-file= *# Name of a specific file that when changed will trigger the restart check. If not specified any classpath file change will trigger the restart.*

*# REMOTE DEVTOOLS (*[RemoteDevToolsProperties](https://github.com/spring-projects/spring-boot/tree/v1.4.4.RELEASE/spring-boot-devtools/src/main/java/org/springframework/boot/devtools/autoconfigure/RemoteDevToolsProperties.java))

spring.devtools.remote.context-path=/.~~spring-boot!~ *# Context path used to handle the remote connection.*

spring.devtools.remote.debug.enabled=true *# Enable remote debug support.*

spring.devtools.remote.debug.local-port=8000 *# Local remote debug server port.*

spring.devtools.remote.proxy.host= *# The host of the proxy to use to connect to the remote application.*

spring.devtools.remote.proxy.port= *# The port of the proxy to use to connect to the remote application.*

spring.devtools.remote.restart.enabled=true *# Enable remote restart.*

spring.devtools.remote.secret= *# A shared secret required to establish a connection (required to enable remote support).*

spring.devtools.remote.secret-header-name=X-AUTH-TOKEN *# HTTP header used to transfer the shared secret.*

# 读取配置文件

springboot中如果需要一些配置的话，可以在工程的src/main/resource下的application.properties文件中进行配置。

在工程的src/main/resources 下修改核心配置文application.properties, 添加内容如下：

name=传智播客

url=http://www.itcast.cn

## 读取核心配置文件一

在Controller中添加：

@RestController

**public** **class** HelloWorldController {

@Resource

**private** Environment environment;

@RequestMapping("info")

**public** String info() {

System.***out***.println(environment.getProperty("name"));

System.***out***.println(environment.getProperty("url"));

**return** "Hello world!";

}

}

就可以直接把配置文件信息打印出来。

注意包名是：org.springframework.core.env.Environment

## 读取核心配置文件二

还是上面的例子，我们只需要是一个@value注解即可获取配置文件内容

@Value("${name}")

**private** String name;

@Value("${url}")

**private** String url;