spring-boot总结

笔记本：	spring_boot		
创建时间：	2018/9/16/周日 9:50	**更新时间：**	2018/9/16/周日 10:20
作者：	1634896520@qq.com		
URL：	https://www.cnblogs.com/huangjianping/p/8203811.html		

spring-boot总结

spring-boot以pom.xml依赖的方式免去了原来繁琐的配置整合，默认提供一些功能的配置，强大的集成和解耦其他框架的作用，可以将一些想要的功能用一些规则的手段来实现。它可以帮助开发者快速并且简单的构建项目，大多数srping-boot项目只需要很少的配置文件。
快速构建spring-boot项目： http://start.spring.io;

一、SpringBoot核心功能

1、独立运行Spring项目
Spring boot 可以以jar包形式独立运行，运行一个Spring Boot项目只需要通过java -jar xx.jar来运行。

2、内嵌servlet容器
Spring Boot可以选择内嵌Tomcat、jetty或者Undertow,这样我们无须以war包形式部署项目。

3、提供starter简化Maven配置
spring提供了一系列的start pom来简化Maven的依赖加载，例如，当你使用了spring-boot-starter-web，会自动加入依赖包。

4、自动装配Spring
SpringBoot会根据在类路径中的jar包，类、为jar包里面的类自动配置Bean，这样会极大地减少我们要使用的配置。当然，SpringBoot只考虑大多数的开发场景，并不是所有的场景，若在实际开发中我们需要配置Bean，而SpringBoot没有提供支持，则可以自定义自动配置。

5、准生产的应用监控
SpringBoot提供基于http ssh telnet对运行时的项目进行监控。

6、无代码生产和xml配置
SpringBoot不是借助与代码生成来实现的，而是通过条件注解来实现的，这是Spring4.x提供的新特性。

二、SpringBoot几个常用的注解
（1）@RestController和@Controller指定一个类，作为控制器的注解
（2）@RequestMapping方法级别的映射注解，这一个用过Spring MVC的小伙伴相信都很熟悉
（3）@EnableAutoConfiguration和@SpringBootApplication是类级别的注解，根据maven依赖的jar来自动猜测完成正确的spring的对应配置，只要引入了spring-boot-starter-web的依赖，默认会自动配置Spring MVC和tomcat容器
（4）@Configuration类级别的注解，一般这个注解，我们用来标识main方法所在的类,完成元数据bean的初始化。
（5）@ComponentScan类级别的注解，自动扫描加载所有的Spring组件包括Bean注入，一般用在main方法所在的类上
（6）@ImportResource类级别注解，当我们必须使用一个xml的配置时，使用@ImportResource和@Configuration来标识这个文件资源的类。
（7）@Autowired注解，一般结合@ComponentScan注解，来自动注入一个Service或Dao级别的Bean

（8）@Component类级别注解，用来标识一个组件，比如我自定了一个filter，则需要此注解标识之后，Spring Boot才会正确识别。

三、推荐spring-boot集成的一些功能 网站

https://www.jianshu.com/p/9a08417e4e84

四、常见面试题

1.什么是Spring Boot？

多年来，随着新功能的增加，spring变得越来越复杂。只需访问https://spring.io/projects页面，我们就会看到可以在我们的应用程序中使用的所有Spring项目的不同功能。

如果必须启动一个新的Spring项目，我们必须添加构建路径或添加Maven依赖关系，配置应用程序服务器，添加spring配置。

因此，开始一个新的spring项目需要很多努力，因为我们现在必须从头开始做所有事情。

Spring Boot是解决这个问题的方法。Spring Boot已经建立在现有spring框架之上。使用spring启动，我们避免了之前我们必须做的所有样板代码和配置。

因此，Spring Boot可以帮助我们以最少的工作量，更加健壮地使用现有的Spring功能。

2.Spring Boot有哪些优点？

- 减少开发，测试时间和努力。
- 使用JavaConfig有助于避免使用XML。
- 避免大量的Maven导入和各种版本冲突。
- 提供意见发展方法。
- 通过提供默认值快速开始开发。
- 没有单独的Web服务器需要。这意味着你不再需要启动Tomcat，Glassfish或其他任何东西。
- 需要更少的配置 因为没有web.xml文件。只需添加用@ Configuration注释的类，然后添加用@Bean注释的方法，Spring将自动加载对象并像以前一样对其进行管理。您甚至可以将@Autowired添加到bean方法中，以使Spring自动装入需要的依赖关系中。
- 基于环境的配置 使用这些属性，您可以将您正在使用的环境传递到应用程序：-Dspring.profiles.active = {enviornment}。在加载主应用程序属性文件后，Spring将在（application{environment} .properties）中加载后续的应用程序属性文件。

3.什么是JavaConfig？

Spring JavaConfig是Spring社区的产品，它提供了配置Spring IoC容器的纯Java方法。因此它有助于避免使用XML配置。使用JavaConfig的优点在于：

面向对象的配置。由于配置被定义为JavaConfig中的类，因此用户可以充分利用Java中的面向对象功能。一个配置类可以继承另一个，重写它的@Bean方法等。

减少或消除XML配置。基于依赖注入原则的外化配置的好处已被证明。但是，许多开发人员不希望在XML和Java之间来回切换。

JavaConfig为开发人员提供了一种纯Java方法来配置与XML配置概念相似的Spring容器。

从技术角度来讲，只使用JavaConfig配置类来配置容器是可行的，但实际上很多人认为将JavaConfig与XML混合匹配是理想的。

类型安全和重构友好。JavaConfig提供了一种类型安全的方法来配置Spring容器。由于Java 5.0对泛型的支持，现在可以按类型而不是按名称检索bean，不需要任何强制转换或基于字符串的查找。

4.如何重新加载Spring Boot上的更改，而无需重新启动服务器？

这可以使用DEV工具来实现。通过这种依赖关系，您可以节省任何更改，嵌入式tomcat将重新启动。

Spring Boot有一个开发工具（DevTools）模块，它有助于提高开发人员的生产力。Java开发人员面临的一个主要挑战是将文件更改自动部署到服务器并自动重启服务器。

开发人员可以重新加载Spring Boot上的更改，而无需重新启动服务器。这将消除每次手动部署更改的需要。Spring Boot在发布它的第一个版本时没有这个功能。

这是开发人员最需要的功能。DevTools模块完全满足开发人员的需求。该模块将在生产环境中被禁用。它还提供H2数据库控制台以更好地测试应用程序。

```
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-devtools</artifactId>
<optional>true</optional>
</dependency>
```

5.Spring Boot中的监视器是什么？

Spring boot actuator是spring启动框架中的重要功能之一。Spring boot监视器可帮助您访问生产环境中正在运行的应用程序的当前状态。

有几个指标必须在生产环境中进行检查和监控。即使一些外部应用程序可能正在使用这些服务来向相关人员触发警报消息。监视器模块公开了一组可直接作为HTTP URL访问的REST端点来检查状态。

6.如何在Spring Boot中禁用Actuator端点安全性？

默认情况下，所有敏感的HTTP端点都是安全的，只有具有ACTUATOR角色的用户才能访问它们。

安全性是使用标准的HttpServletRequest.isUserInRole方法实施的。 我们可以使用management.security.enabled = false 来禁用安全性。只有在执行机构端点在防火墙后访问时，才建议禁用安全性。

如何在自定义端口上运行Spring Boot应用程序？

为了在自定义端口上运行Spring Boot应用程序，您可以在application.properties中指定端口。

server.port = 8090

7.什么是YAML？

YAML是一种人类可读的数据序列化语言。它通常用于配置文件。

与属性文件相比，如果我们想要在配置文件中添加复杂的属性，YAML文件就更加结构化，而且更少混淆。可以看出YAML具有分层配置数据。

8.如何实现Spring Boot应用程序的安全性？

为了实现Spring Boot的安全性，我们使用 spring-boot-starter-security依赖项，并且必须添加安全配置。它只需要很少的代码。配置类将必须扩展WebSecurityConfigurerAdapter并覆盖其方法。

9.如何使用Spring Boot实现分页和排序？

使用Spring Boot实现分页非常简单。使用Spring Data-JPA可以实现将可分页的org.springframework.data.domain.Pageable传递给存储库方法。

10.springboot常用的starter有哪些

spring-boot-starter-web嵌入tomcat和webkaifa需要servlet与jsp支持

spring-boot-starter-data-jpa数据库支持

spring-boot-starter-data-redis数据库支持

spring-boot-starter-data-solr solr支持

mybatis-spring-boot-starter第三方的mybatis集成starter

11·springboot自动配置的原理

在spring程序main方法中添加@SpringBootApplication或者@EnableAutoConfiguration

会自动去mavenzhong读取每个starter中的spring.factories文件，该文件配置了所有需要被创建spring容器中的bean

12·springboot读取配置文件的方式

springboot默认读取配置文件为application.properties或者application.yml

13·如何使用springboot部署到不同的服务器

1、在一个项目中生成一个war文件。

2、将它部署到想要部署的服务器中（websphere或者weblogic或者Tomcat and so on）

14·Spring Boot Starter的面试题

1.常见的starter会包几个方面的内容。分别是什么。

// 常见的starter会包括下面四个方面的内容 // 自动配置文件，根据classpath是否存在指定的类来决定是否要执行该功能的自动配置。 // spring.factories，非常重要，指导Spring Boot找到指定的自动配置文件。 // endpoint：可以理解为一个admin，包含对服务的描述、界面、交互(业务信息的查询)。 // health indicator:该starter提供的服务的健康指标。 两个需要注意的点： // 1. @ConditionalOnMissingBean的作用是：只有对应的bean在系统中都没有被创建，它修饰的初始化代码块才会执行，【用户自己手动创建的bean优先】。 // 2. Spring Boot Starter找到自动配置文件(xxxxAutoConfiguration之类的文件)的方式有两种： // spring.factories:由Spring Boot触发探测classpath目录下的类，进行自动配置； // @EnableXxxxx:有时需要由starter的用户触发*查找自动配置文件的过程

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

2.总结Spring Boot Starter的工作原理

// Spring Boot Starter的工作原理如下： // 1. Spring Boot 在启动时扫描项目所依赖的JAR包，寻找包含spring.factories文件的JAR // 2. 根据spring.factories配置加载AutoConfigure类 // 3. 根据 @Conditional注解的条件，进行自动配置并将Bean注入Spring Context

1 2 3 4 5 6 7

3.谈谈你对Spring Boot的认识。

// spring Boot是一个开源框架，它可用于创建可执行的Spring应用程序，采用了习惯优于配置的方法。此框架的神奇之处在于@EnableAutoConfiguration注解，此注释自动载入应用程序所需的所有Bean——这依赖于Spring Boot在类路径中的查找。 1. @Enable*注释 @Enable*注释并不是新发明的注释，早在Spring 3框架就引入了这些注释，用这些注释替代XML配置文件。 很多Spring开发者都知道@EnableTransactionManagement注释，它能够声明事务管理；@EnableWebMvc注释，它能启用Spring MVC；以及@EnableScheduling注释，它可以初始化一个调度器。 2. 属性映射 下面看MongoProperties类，它是一个Spring Boot属性映射的例子： @ConfigurationProperties(prefix = "spring.data.mongodb") public class MongoProperties { private String host; private int port = DBPort.PORT; private String uri = "mongodb://localhost/test"; private String database; // ... getters/ setters omitted } @ConfigurationProperties注释将POJO关联到指定前缀的每一个属性。例如，spring.data.mongodb.port属性将映射到这个类的端口属性。 强烈建议Spring Boot开发者使用这种方式来删除与配置属性相关的瓶颈代码。 3.@Conditional注释 Spring Boot的强大之处在于使用了Spring 4框架的新特性：@Conditional注释，此注释使得只有在特定条件满足时才启用一些配置。 在Spring Boot的org.springframework.boot.autoconfigure.condition包中说明了使用@Conditional注释能给我们带来什么，下面对这些注释做一个概述： @ConditionalOnBean @ConditionalOnClass @ConditionalOnExpression @ConditionalOnMissingBean @ConditionalOnMissingClass @ConditionalOnNotWebApplication @ConditionalOnResource @ConditionalOnWebApplication 4.应用程序上下文初始化器 spring.factories还提供了第二种可能性，即定义应用程序的初始化。这使得我们可以在应用程序载入前操纵Spring的应用程序上下文ApplicationContext。 特别是，可以在上下文创建监听器，使用ConfigurableApplicationContext类的addApplicationListener()方法。 AutoConfigurationReportLoggingInitializer监听到系统事件时，比如上下文刷新或应用程序启动故障之类的事件，Spring Boot可以执行一些工作。这有助于我们以调试模式启动应用程序时创建自动配置的报告。 要以调试模式启动应用程序，可以使用-Ddebug标识，或者在application.properties文件这添加属性debug= true。

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46

4.自定义springboot-starter注意事项

// 1. springboot默认scan的包名是其main类所在的包名。如果引入的starter包名不一样，需要自己添加scan。 @ComponentScan(basePackages = {"com.xixicat.demo","com.xixicat.sms"}) // 2. 对于starter中有feign的，需要额外指定 @EnableFeignClients(basePackages = {"com.xixicat.sms"}) // 3. 对于exclude一些autoConfig @EnableAutoConfiguration(exclude = {MetricFilterAutoConfiguration.class})