Spring Boot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、为什么要用SpringBoot

Spring Boot优点非常多，如：

一、独立运行

Spring Boot而且内嵌了各种servlet容器，Tomcat、Jetty等，现在不再需要打成war包部署到容器中，Spring Boot只要打成一个可执行的jar包就能独立运行，所有的依赖包都在一个jar包内。

二、简化配置

spring-boot-starter-web启动器自动依赖其他组件，简少了maven的配置。

三、自动配置

Spring Boot能根据当前类路径下的类、jar包来自动配置bean，如添加一个spring-boot-starter-web启动器就能拥有web的功能，无需其他配置。

四、无代码生成和XML配置

Spring Boot配置过程中无代码生成，也无需XML配置文件就能完成所有配置工作，这一切都是借助于条件注解完成的，这也是Spring4.x的核心功能之一。

五、应用监控

Spring Boot提供一系列端点可以监控服务及应用，做健康检测

3、Spring Boot 有哪些优点？

Spring Boot 的优点有：
1、减少开发，测试时间和努力。
2、使用 JavaConfig 有助于避免使用 XML。
3、避免大量的 Maven 导入和各种版本冲突。
4、提供意见发展方法。
5、通过提供默认值快速开始开发。
6、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。
7、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。
8、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。

4、Spring Boot 的核心注解是哪个？它主要由哪几个注解组成的？

启动类上面的注解是@SpringBootApplication，它也是 Spring Boot 的核心注解，主要组合包含了以下 3 个注解：
@SpringBootConfiguration：组合了 @Configuration 注解，实现配置文件的功能。
@EnableAutoConfiguration：打开自动配置的功能，也可以关闭某个自动配置的选项，如关闭数据源自动配置功能： @SpringBootApplication(exclude = { DataSourceAutoConfiguration.class })。
@ComponentScan：Spring组件扫描

5、运行Spring Boot有哪几种方式

1）打包用命令或者放到容器中运行
2）用 Maven/Gradle 插件运行
3）直接执行 main 方法运行

6、如何理解 Spring Boot 中的 Starters？

Starters是什么：
Starters可以理解为启动器，它包含了一系列可以集成到应用里面的依赖包，你可以一站式集成Spring及其他技术，而不需要到处找示例代码和依赖包。如你想使用Spring JPA访问数据库，只要加入springboot-starter-data-jpa启动器依赖就能使用了。Starters包含了许多项目中需要用到的依赖，它们能快速持续的运行，都是一系列得到支持的管理传递性依赖。
Starters命名：
Spring Boot官方的启动器都是以spring-boot-starter-命名的，代表了一个特定的应用类型。第三方的启动器不能以spring-boot开头命名，它们都被Spring Boot官方保留。一般一个第三方的应该这样命名，像mybatis的mybatis-spring-boot-starter。
Starters分类：

1. Spring Boot应用类启动器

启动器名称	功能描述
spring-boot-starter	包含自动配置、日志、YAML的支持。
spring-boot-starter-web	使用Spring MVC构建web 工程，包含restful，默认使用Tomcat容器。
...	...

2.Spring Boot生产启动器

启动器名称	功能描述
spring-boot-starter-actuator	提供生产环境特性，能监控管理应用。

3. Spring Boot技术类启动器

启动器名称	功能描述
spring-boot-starter-json	提供对JSON的读写支持。
spring-boot-starter-logging	默认的日志启动器，默认使用Logback。
...	...

4. 其他第三方启动器

7、如何在Spring Boot启动的时候运行一些特定的代码？

如果你想在Spring Boot启动的时候运行一些特定的代码，你可以实现接口ApplicationRunner或者CommandLineRunner，这两个接口实现方式一样，它们都只提供了一个run方法。
CommandLineRunner：启动获取命令行参数

8、Spring Boot 需要独立的容器运行吗？

可以不需要，内置了 Tomcat/ Jetty 等容器

9、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

10、如何使用Spring Boot实现异常处理？

Spring提供了一种使用ControllerAdvice处理异常的非常有用的方法。 我们通过实现一个ControlerAdvice类，来处理控制器类抛出的所有异常

11、你如何理解 Spring Boot 中的 Starters

Starters可以理解为启动器，它包含了一系列可以集成到应用里面的依赖包，你可以一站式集成 Spring 及其他技术，而不需要到处找示例代码和依赖包。如你想使用 Spring JPA 访问数据库，只要加入 spring-boot-starter-data-jpa 启动器依赖就能使用了

12、springboot常用的starter有哪些

spring-boot-starter-web 嵌入tomcat和web开发需要servlet与jsp支持
spring-boot-starter-data-jpa 数据库支持
spring-boot-starter-data-redis 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

13、SpringBoot 实现热部署有哪几种方式

主要有两种方式：
Spring Loaded
Spring-boot-devtools

14、如何理解 Spring Boot 配置加载顺序

在 Spring Boot 里面，可以使用以下几种方式来加载配置。
1）properties文件；
2）YAML文件；
3）系统环境变量；

4）命令行参数；
等等……

15、Spring Boot 的核心配置文件有哪几个？它们的区别是什么？

Spring Boot 的核心配置文件是 application 和 bootstrap 配置文件。
application 配置文件这个容易理解，主要用于 Spring Boot 项目的自动化配置。
bootstrap 配置文件有以下几个应用场景。

1. 使用 Spring Cloud Config 配置中心时，这时需要在 bootstrap 配置文件中添加连接到配置中心的配置属性来加载外部配置中心的配置信息；
2. 一些固定的不能被覆盖的属性；
3. 一些加密/解密的场景

16、如何集成 Spring Boot 和 ActiveMQ

对于集成 Spring Boot 和 ActiveMQ，我们使用spring-boot-starter-activemq依赖关系。 它只需要很少的配置，并且不需要样板代码

17、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：

1、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。

2、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。

3、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。

18、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>
```

19、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。

20、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。

21、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
```

22、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。

23、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。

24、如何集成 Spring Boot 和 ActiveMQ？

对于集成 Spring Boot 和 ActiveMQ，我们使用
依赖关系。 它只需要很少的配置，并且不需要样板代码。

25、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传递给存储库方法。

26、什么是 Swagger？你用 Spring Boot 实现了它吗？

Swagger 广泛用于可视化 API，使用 Swagger UI 为前端开发人员提供在线沙箱。Swagger 是用于生成 RESTful Web 服务的可视化表示的工具，规范和完整框架实现。它使文档能够以与服务器相同的速度更新。当通过 Swagger 正确定义时，消费者可以使用最少量的实现逻辑来理解远程服务并与其进行交互。因此，Swagger
消除了调用服务时的猜测。

27、什么是 Spring Profiles？

Spring Profiles 允许用户根据配置文件（dev，test，prod 等）来注册 bean。因此，当应用程序在开发中运行时，只有某些 bean 可以加载，而在 PRODUCTION中，某些其他 bean 可以加载。假设我们的要求是 Swagger 文档仅适用于 QA 环境，并且禁用所有其他文档。这可以使用配置文件来完成。Spring Boot 使得使用配置文件非常简单。

28、什么是 Spring Batch？

Spring Boot Batch 提供可重用的函数，这些函数在处理大量记录时非常重要，包括日志/跟踪，事务管理，作业处理统计信息，作业重新启动，跳过和资源管理。它还提供了更先进的技术服务和功能，通过优化和分区技术，可以实现极高批量和高性能批处理作业。简单以及复杂的大批量批处理作业可以高度可扩展的方式利用框架处理重要大量的信息。

29、什么是 FreeMarker 模板？

FreeMarker 是一个基于 Java 的模板引擎，最初专注于使用 MVC 软件架构进行动态网页生成。使用 Freemarker 的主要优点是表示层和业务层的完全分离。程序员可以处理应用程序代码，而设计人员可以处理 html 页面设计。最后使用freemarker 可以将这些结合起来，给出最终的输出页面。

30、如何使用 Spring Boot 实现异常处理？

Spring 提供了一种使用 ControllerAdvice 处理异常的非常有用的方法。 我们通过实现一个 ControlerAdvice 类，来处理控制器类抛出的所有异常。

31、您使用了哪些 starter maven 依赖项？

使用了下面的一些依赖项

```
spring-boot-starter-activemq
spring-boot-starter-security
```

这有助于增加更少的依赖关系，并减少版本的冲突。

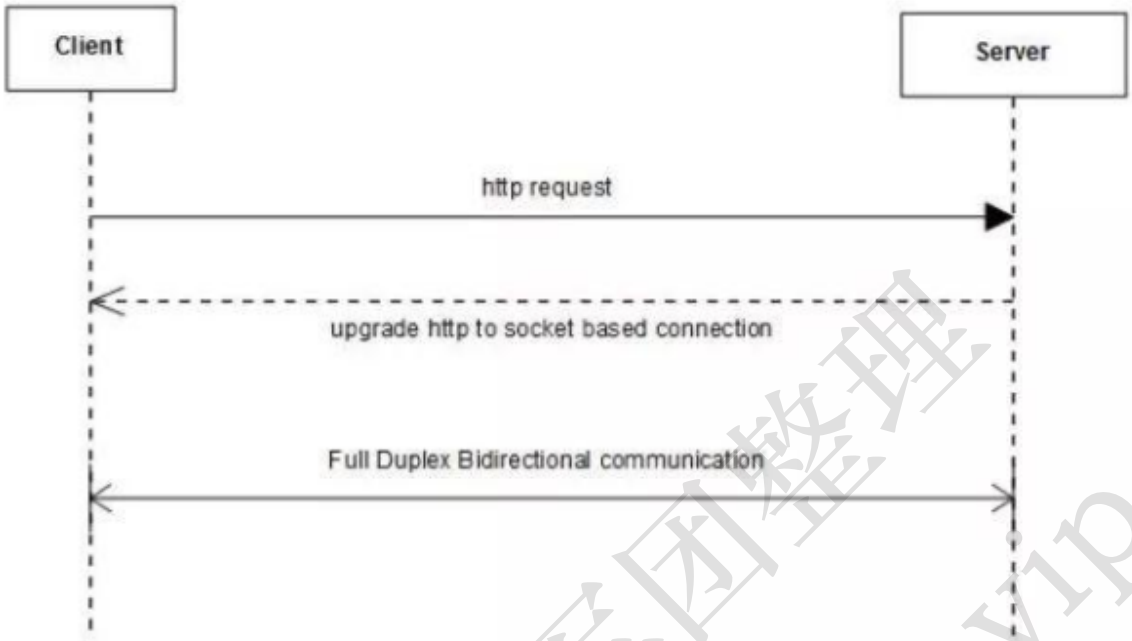
32、什么是 CSRF 攻击？

CSRF 代表跨站请求伪造。这是一种攻击，迫使最终用户在当前通过身份验证的Web 应用程序上执行不需要的操作。CSRF 攻击专门针对状态改变请求，而不是数据窃取，因为攻击者无法查看对伪造请求的响应。

这有助于增加更少的依赖关系，并减少版本的冲突。

33、什么是 WebSockets？

WebSocket 是一种计算机通信协议，通过单个 TCP 连接提供全双工通信信道。



1、WebSocket 是双向的 -使用 WebSocket 客户端或服务器可以发起消息发送。
2、WebSocket 是全双工的 -客户端和服务器通信是相互独立的。
3、单个 TCP 连接 -初始连接使用 HTTP，然后将此连接升级到基于套接字的连接。然后这个单一连接用于所有未来的通信
4、Light -与 http 相比，WebSocket 消息数据交换要轻得多。

34、什么是 AOP？

在软件开发过程中，跨越应用程序多个点的功能称为交叉问题。这些交叉问题与应用程序的主要业务逻辑不同。因此，将这些横切关注与业务逻辑分开是面向方面编程（AOP）的地方。

35、什么是 Apache Kafka？

Apache Kafka 是一个分布式发布 - 订阅消息系统。它是一个可扩展的，容错的发布 - 订阅消息系统，它使我们能够构建分布式应用程序。这是一个 Apache 顶级项目。Kafka 适合离线和在线消息消费。

36、我们如何监视所有 Spring Boot 微服务？

Spring Boot 提供监视器端点以监控各个微服务的度量。这些端点对于获取有关应用程序的信息（如它们是否已启动）以及它们的组件（如数据库等）是否正常运行很有帮助。但是，使用监视器的一个主要缺点或困难是，我们必须单独打开应用程序的知识点以了解其状态或健康状况。想象一下涉及 50 个应用程序的微服务，管理员将不得不击中所有 50 个应用程序的执行终端。为了帮助我们处理这种情况，我们将使用位于的开源项目。 它建立在 Spring Boot Actuator 之上，它提供了一个 Web UI，使我们能够可视化多个应用程序的度量。

37、Spring Boot 的配置文件有哪几种格式？它们有什么区别？

.properties 和 .yml，它们的区别主要是书写格式不同。

1).properties

```
app.user.name = javastack
```

2).yml

```
app:
  user:
    name: javastack
```

另外，.yml 格式不支持 `@PropertySource` 注解导入配置。

38、开启 Spring Boot 特性有哪几种方式？

1）继承spring-boot-starter-parent项目

2）导入spring-boot-dependencies项目依赖

39、Spring Boot 的目录结构是怎样的？

```
cn
 +- javastack
     +- MyApplication.java
     |
     +- customer
     |   +- Customer.java
     |   +- CustomerController.java
     |   +- CustomerService.java
     |   +- CustomerRepository.java
     |
     +- order
         +- Order.java
         +- OrderController.java
         +- OrderService.java
         +- OrderRepository.java
```

这个目录结构是主流及推荐的做法，而在主入口类上加上 @SpringBootApplication 注解来开启 Spring Boot 的各项能力，如自动配置、组件扫描等。

40、运行 Spring Boot 有哪几种方式？

1）打包用命令或者放到容器中运行

2）用 Maven/ Gradle 插件运行

3）直接执行 main 方法运行

41、Spring Boot 自动配置原理是什么？

注解 @EnableAutoConfiguration, @Configuration, @ConditionalOnClass 就是自动配置的核心，首先它得是一个配置文件，其次根据类路径下是否有这个类去自动配置。

42、如何在 Spring Boot 启动的时候运行一些特定的代码？

可以实现接口 ApplicationRunner 或者 CommandLineRunner，这两个接口实现方式一样，它们都只提供了一个 run 方法

43、Spring Boot 有哪几种读取配置的方式？

Spring Boot 可以通过 @PropertySource,@Value,@Environment, @ConfigurationProperties 来绑定变量

44、Spring Boot 支持哪些日志框架？推荐和默认的日志框架是哪个？

Spring Boot 支持 Java Util Logging, Log4j2, Lockback 作为日志框架，如果你使用 Starters 启动器，Spring Boot 将使用 Logback 作为默认日志框架

45、Spring Boot 如何定义多套不同环境配置？

提供多套配置文件，如：

```
applcation.properties

application-dev.properties

application-test.properties

application-prod.properties
```

运行时指定具体的配置文件

46、Spring Boot 可以兼容老 Spring 项目吗，如何做？

可以兼容，使用 `@ImportResource` 注解导入老 Spring 项目配置文件。

47、保护 Spring Boot 应用有哪些方法？

- 在生产中使用HTTPS
- 使用Snyk检查你的依赖关系
- 升级到最新版本
- 启用CSRF保护

- 使用内容安全策略防止XSS攻击
- ...

48、Spring Boot 2.X 有什么新特性？与 1.X 有什么区别？

- 配置变更
- JDK 版本升级
- 第三方类库升级
- 响应式 Spring 编程支持
- HTTP/2 支持
- 配置属性绑定
- 更多改进与加强...

49、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。

org.springframework.boot
spring-boot-devtools
true

50、springboot集成mybatis的过程

添加mybatis的starter maven依赖

org.mybatis.spring.boot
mybatis-spring-boot-starter
1.2.0

在mybatis的接口中 添加@Mapper注解
在application.yml配置数据源信息

51、Spring Boot、Spring MVC 和 Spring 有什么区别？

SpringFrame

> SpringFramework 最重要的特征是依赖注入。所有 SpringModules 不是依赖注入就是 IOC 控制反转。

当我们恰当的使用 DI 或者是 IOC 的时候，我们可以开发松耦合应用。松耦合应用的单元测试可以很容易的进行。

SpringMVC

Spring MVC 提供了一种分离式的方法来开发 Web 应用。通过运用像 DispatcherServelet，MoudlAndView 和 ViewResolver 等一些简单的概念，开发 Web 应用将会变的非常简单。

SpringBoot

Spring 和 SpringMVC 的问题在于需要配置大量的参数。

```
<bean
 class="org.springframework.web.servlet.view.InternalResourceViewResolver">
<property name="prefix">
        <value>/WEB-INF/views/</value>
    </property>
    <property name="suffix">
        <value>.jsp</value>
    </property>
  </bean>

  <mvc:resources mapping="/webjars/**" location="/webjars/"/>
```

Spring Boot 通过一个自动配置和启动的项来目解决这个问题。为了更快的构建产品就绪应用程序，Spring Boot 提供了一些非功能性特征。

52、什么是 Spring Boot Stater ？

> 启动器是一套方便的依赖没描述符，它可以放在自己的程序中。你可以一站式的获取你所需要的 Spring 和相关技术，而不需要依赖描述符的通过示例代码搜索和复制黏贴的负载。

例如，如果你想使用 Sping 和 JPA 访问数据库，只需要你的项目包含 spring-boot-starter-data-jpa 依赖项，你就可以完美进行。

问题四 **你能否举一个例子来解释更多 Staters 的内容？**

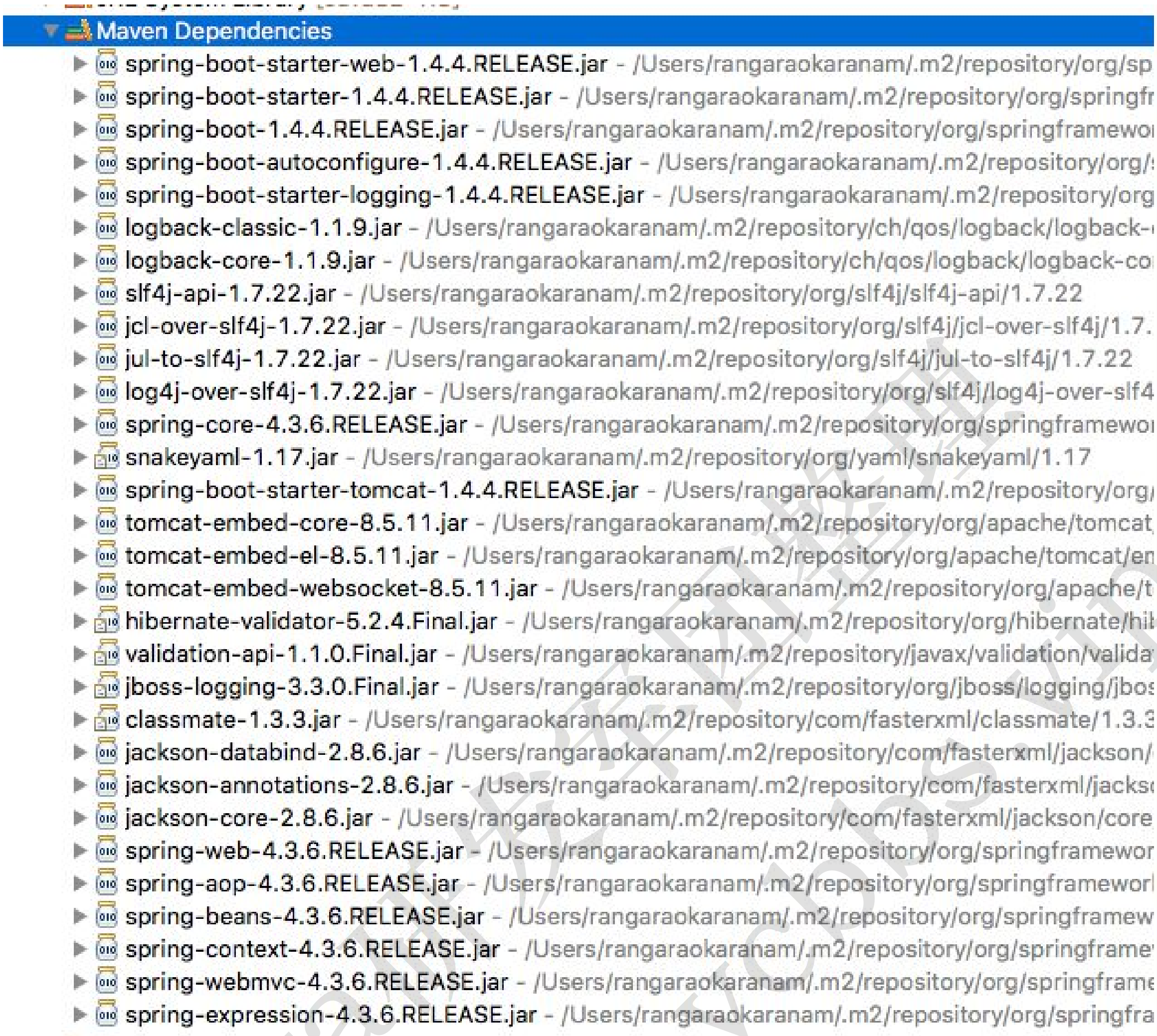
让我们来思考一个 Stater 的例子 -Spring Boot Stater Web。

如果你想开发一个 web 应用程序或者是公开 REST 服务的应用程序。Spring Boot Start Web 是首选。让我们使用 Spring Initializr 创建一个 Spring Boot Start Web 的快速项目。

Spring Boot Start Web 的依赖项

```
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-web</artifactId>
</dependency>
```

下面的截图是添加进我们应用程序的不同的依赖项



依赖项可以被分为

```
Spring - core, beans, context, aop

Web MVC - (Spring MVC)

Jackson - for JSON Binding

Validation - Hibernate,Validation API

Enbedded Servlet Container - Tomcat

Logging - logback,slf4j
```

任何经典的 Web 应用程序都会使用所有这些依赖项。Spring Boot Starter Web 预先打包了这些依赖项。

> 作为一个开发者，我不需要再担心这些依赖项和它们的兼容版本。

53、Spring Boot 还提供了其它的哪些 Starter Project Options？

Spring Boot 也提供了其它的启动器项目包括，包括用于开发特定类型应用程序的典型依赖项。

spring-boot-starter-web-services - SOAP Web Services

spring-boot-starter-web - Web 和 RESTful 应用程序

spring-boot-starter-test - 单元测试和集成测试

spring-boot-starter-jdbc - 传统的 JDBC

spring-boot-starter-hateoas - 为服务添加 HATEOAS 功能

spring-boot-starter-security - 使用 SpringSecurity 进行身份验证和授权

spring-boot-starter-data-jpa - 带有 Hibeernate 的 Spring Data JPA

spring-boot-starter-data-rest - 使用 Spring Data REST 公布简单的 REST 服务

54、Spring 是如何快速创建产品就绪应用程序的？

Spring Boot 致力于快速产品就绪应用程序。为此，它提供了一些譬如高速缓存，日志记录，监控和嵌入式服务器等开箱即用的非功能性特征。

spring-boot-starter-actuator - 使用一些如监控和跟踪应用的高级功能

spring-boot-starter-undertow, spring-boot-starter-jetty, spring-boot-starter-tomcat - 选择您的特定嵌入式 Servlet 容器

spring-boot-starter-logging - 使用 logback 进行日志记录

spring-boot-starter-cache - 启用 Spring Framework 的缓存支持

###Spring2 和 Spring5 所需要的最低 Java 版本是什么？

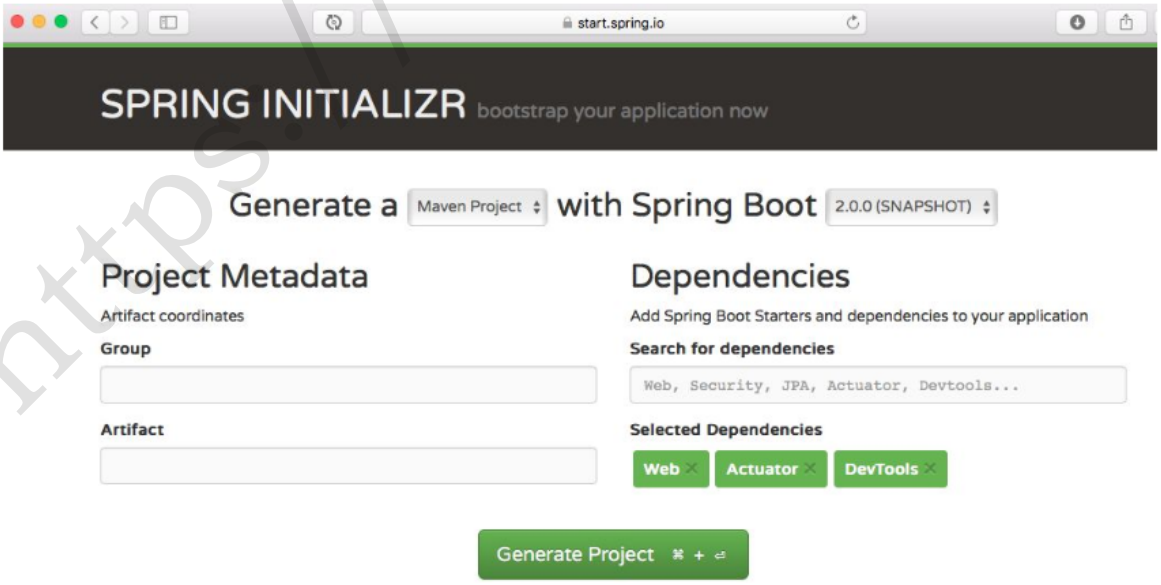
Spring Boot 2.0 需要 Java8 或者更新的版本。Java6 和 Java7 已经不再支持。

推荐阅读：

https://github.com/spring-projects/spring-boot/wiki/Spring-Boot-2.0.0-M1-Release-Notes

55、创建一个 Spring Boot Project 的最简单的方法是什么？

Spring Initializr是启动 Spring Boot Projects 的一个很好的工具。



- 就像上图中所展示的一样，我们需要做一下几步：
- 登录 Spring Initializr，按照以下方式进行选择：
- 选择 com.in28minutes.springboot 为组

- 选择 studet-services 为组件
- 选择下面的依赖项
- Web
- Actuator
- DevTools
- 点击生 GenerateProject
- 将项目导入 Eclipse。文件 - 导入 - 现有的 Maven 项目

56、Spring Initializr 是创建 Spring Boot Projects 的唯一方法吗？

不是的。

Spring Initiatlizr 让创建 Spring Boot 项目变的很容易，但是，你也可以通过设置一个 maven 项目并添加正确的依赖项来开始一个项目。

在我们的 Spring 课程中，我们使用两种方法来创建项目。

第一种方法是 start.spring.io 。
另外一种方法是在项目的标题为“Basic Web Application”处进行手动设置。

手动设置一个 maven 项目

这里有几个重要的步骤：

- 在 Eclipse 中，使用文件 - 新建 Maven 项目来创建一个新项目
- 添加依赖项。
- 添加 maven 插件。
- 添加 Spring Boot 应用程序类。

到这里，准备工作已经做好！

57、如何使用 SpringBoot 自动重装我的应用程序？

使用 Spring Boot 开发工具。

把 Spring Boot 开发工具添加进入你的项目是简单的。

把下面的依赖项添加至你的 Spring Boot Project pom.xml 中

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <scope>runtime</scope>
</dependency>
```

重启应用程序，然后就可以了。

同样的，如果你想自动装载页面，有可以看看 FiveReload

- http://www.logicbig.com/tutorials/spring-framework/spring-boot/boot-live-reload/.

在我测试的时候，发现了 LiveReload 漏洞，如果你测试时也发现了，请一定要告诉我们。

58、 什么是嵌入式服务器？我们为什么要使用嵌入式服务器呢?

思考一下在你的虚拟机上部署应用程序需要些什么。

第一步： 安装 Java

第二部： 安装 Web 或者是应用程序的服务器（Tomat/Wbesphere/Weblogic 等等）

第三部： 部署应用程序 war 包

如果我们想简化这些步骤，应该如何做呢？

让我们来思考如何使服务器成为应用程序的一部分？

> 你只需要一个安装了 Java 的虚拟机，就可以直接在上面部署应用程序了，
> 是不是很爽？

这个想法是嵌入式服务器的起源。

当我们创建一个可以部署的应用程序的时候，我们将会把服务器（例如，tomcat）嵌入到可部署的服务器中。

> 例如，对于一个 Spring Boot 应用程序来说，你可以生成一个包含 Embedded Tomcat 的应用程序 jar。你就可以想运行正常 Java 应用程序一样来运行 web 应用程序了。

嵌入式服务器就是我们的可执行单元包含服务器的二进制文件（例如，tomcat.jar）。

59、如何在 Spring Boot 中添加通用的 JS 代码？

在源文件夹下，创建一个名为 static 的文件夹。然后，你可以把你的静态的内容放在这里面。

例如，myapp.js 的路径是 resources\static\js\myapp.js

你可以参考它在 jsp 中的使用方法

```
<csript src="/js/myapp.js"></script>
```

错误：HAL browser gives me unauthorized error - Full authenticaition is required to access this resource.

该如何来修复这个错误呢？

```
{
  "timestamp": 1488656019562,
  "status": 401,
  "error": "Unauthorized",
  "message": "Full authentication is required to access this resource.",
  "path": "/beans"
}
```

两种方法：

方法 1：关闭安全验证

application.properties

```
management.security.enabled:FALSE
```

方法二：在日志中搜索密码并传递至请求标头中

60、什么是 Spring Date？

来自：//projects.spring.io/spring- data/

> Spring Data 的使命是在保证底层数据存储特殊性的前提下，为数据访问提供一个熟悉的，一致性的，基于 Spring 的编程模型。这使得使用数据访问技术，关系数据库和非关系数据库，map-reduce 框架以及基于云的数据服务变得很容易。

为了让它更简单一些，Spring Data 提供了不受底层数据源限制的 Abstractions 接口。

下面来举一个例子

```
interface TodoRepository extends CrudRepository<Todo, Long> {
```

你可以定义一简单的库，用来插入，更新，删除和检索代办事项，而不需要编写大量的代码。

61、什么是 Spring Data REST?

Spring Data TEST 可以用来发布关于 Spring 数据库的 HATEOAS RESTful 资源。

下面是一个使用 JPA 的例子

```
@RepositoryRestResource(collectionResourceRel = "todos", path = "todos")
public interface TodoRepository
        extends PagingAndSortingRepository<Todo, Long> {
```

不需要写太多代码，我们可以发布关于 Spring 数据库的 RESTful API。

下面展示的是一些关于 TEST 服务器的例子

POST

- URL:http：//localhost：8080/todos
- Use Header:Content-Type:Type:application/json
- Request Content

代码如下

```
{
  "user": "Jill",
  "desc": "Learn Hibernate",
  "done": false
}
```

响应内容

```
{
  "user": "Jill",
  "desc": "Learn Hibernate",
  "done": false,
  "_links": {
"self": {
  "href": "http://localhost:8080/todos/1"
},
"todo": {
  "href": "http://localhost:8080/todos/1"
}
  }
}
```

响应包含新创建资源的 href。

62、path=”users”, collectionResourceRel=”users” 如何与 Spring Data Rest 一起使用？

```
@RepositoryRestResource(collectionResourceRel = "users", path = "users")

public interface UserRestRepository extends
PagingAndSortingRepository<User, Long>
```

- path- 这个资源要导出的路径段。
- collectionResourceRel- 生成指向集合资源的链接时使用的 rel 值。在生成 HATEOAS 链接时使用。

63、当 Spring Boot 应用程序作为 Java 应用程序运行时，后台会发生什么？

如果你使用 Eclipse IDE，Eclipse maven 插件确保依赖项或者类文件的改变一经添加，就会被编译并在目标文件中准备好！在这之后，就和其它的 Java 应用程序一样了。

当你启动 java 应用程序的时候，spring boot 自动配置文件就会魔法般的启用了。

- 当 Spring Boot 应用程序检测到你正在开发一个 web 应用程序的时候，它就会启动 tomcat。

64、我们能否在 spring-boot-starter-web 中用 jetty 代替 tomcat？

在 spring-boot-starter-web 移除现有的依赖项，并把下面这些添加进去。

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
    <exclusions>
        <exclusion>
            <groupId>org.springframework.boot</groupId>
```

```
            <artifactId>spring-boot-starter-tomcat</artifactId>
        </exclusion>
    </exclusions>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-jetty</artifactId>
</dependency>
```

65、如何使用 Spring Boot 生成一个 WAR 文件？

推荐阅读:

- https://spring.io/guides/gs/convert-jar-to-war/

下面有 spring 说明文档直接的链接地址：

- https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#build-tool-plugins-maven-packaging

66、如何使用 Spring Boot 部署到不同的服务器？

你需要做下面两个步骤：

- 在一个项目中生成一个 war 文件。
- 将它部署到你最喜欢的服务器（websphere 或者 Weblogic 或者 Tomcat and so on）。

第一步：这本入门指南应该有所帮助：
https://spring.io/guides/gs/convert-jar-to-war/

第二步：取决于你的服务器。

67、RequestMapping 和 GetMapping 的不同之处在哪里？

- RequestMapping 具有类属性的，可以进行 GET,POST,PUT 或者其它的注释中具有的请求方法。
- GetMapping 是 GET 请求方法中的一个特例。它只是 ResquestMapping 的一个延伸，目的是为了提高清晰度。

68、为什么我们不建议在实际的应用程序中使用 Spring Data Rest?

我们认为 Spring Data Rest 很适合快速原型制造！在大型应用程序中使用需要谨慎。

通过 Spring Data REST 你可以把你的数据实体作为 RESTful 服务直接发布。

当你设计 RESTful 服务器的时候，最佳实践表明，你的接口应该考虑到两件重要的事情：

- 你的模型范围。
- 你的客户。

通过 With Spring Data REST，你不需要再考虑这两个方面，只需要作为 TEST 服务发布实体。

这就是为什么我们建议使用 Spring Data Rest 在快速原型构造上面，或者作为项目的初始解决方法。对于完整演变项目来说，这并不是一个好的注意。

69、在 Spring Initializer 中，如何改变一个项目的包名字？

好消息是你可以定制它。点击链接“转到完整版本”。你可以配置你想要修改的包名称！

70、可以配置 application.propertierde 的完整的属性列表在哪里可以找到？

这里是完整的指南：

- https://docs.spring.io/spring-boot/docs/current/reference/html/common-application-properties.html

71、JPA 和 Hibernate 有哪些区别？

简而言之

- JPA 是一个规范或者接口
- Hibernate 是 JPA 的一个实现

当我们使用 JPA 的时候，我们使用 javax.persistence 包中的注释和接口时，不需要使用 hibernate 的导入包。

我们建议使用 JPA 注释，因为哦我们没有将其绑定到 Hibernate 作为实现。后来（我知道 - 小于百分之一的几率），我们可以使用另一种 JPA 实现。

72、使用 Spring Boot 启动连接到内存数据库 H2 的 JPA 应用程序需要哪些依赖项？

在 Spring Boot 项目中，当你确保下面的依赖项都在类路里面的时候，你可以加载 H2 控制台。

- web 启动器
- h2
- jpa 数据启动器

其它的依赖项在下面：

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>

<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>

<dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
    <scope>runtime</scope>
```

```
</dependency>
```

需要注意的一些地方：

- 一个内部数据内存只在应用程序执行期间存在。这是学习框架的有效方式。
- 这不是你希望的真是世界应用程序的方式。
- 在问题“如何连接一个外部数据库？”中，我们解释了如何连接一个你所选择的数据库。

73、如何不通过任何配置来选择 Hibernate 作为 JPA 的默认实现？

因为 Spring Boot 是自动配置的。

下面是我们添加的依赖项

```
<dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
      </dependency>
```

spring-boot-stater-data-jpa 对于 Hibernate 和 JPA 有过渡依赖性。

当 Spring Boot 在类路径中检测到 Hibernate 中，将会自动配置它为默认的 JPA 实现。

74、指定的数据库连接信息在哪里？它是如何知道自动连接至 H2 的？

这就是 Spring Boot 自动配置的魔力。

来自：https://docs.spring.io/spring-boot/docs/current/reference/html/using-boot-auto-configuration.html

Spring Boot auto-configuration 试图自动配置你已经添加的基于 jar 依赖项的 Spring 应用程序。比如说，如果 HSQLDBis 存在你的类路径中，并且，数据库连接 bean 还没有手动配置，那么我们可以自动配置一个内存数据库。

进一步的阅读：

http://www.springboottutorial.com/spring-boot-auto-configuration

75、我们如何连接一个像 MSSQL 或者 orcale 一样的外部数据库？

让我们以 MySQL 为例来思考这个问题：

第一步 - 把 mysql 连接器的依赖项添加至 pom.xml

```
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
</dependency>
```

第二步 - 从 pom.xml 中移除 H2 的依赖项

或者至少把它作为测试的范围。

```
<!--
<dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
<scope>test</scope>
</dependency>
-->
```

第三步 - 安装你的 MySQL 数据库

更多的来看看这里 -https://github.com/in28minutes/jpa-with-hibernate#installing-and-setting-up-mysql

第四步 - 配置你的 MySQL 数据库连接

配置 application.properties

```
spring.jpa.hibernate.ddl-auto=none
spring.datasource.url=jdbc:mysql://localhost:3306/todo_example
spring.datasource.username=todouser
spring.datasource.password=YOUR_PASSWORD
```

第五步 - 重新启动，你就准备好了！

就是这么简单！

76、Spring Boot 配置的默认 H2 数据库的名字是上面？为什么默认的数据库名字是 testdb？

在 application.properties 里面，列出了所有的默认值

- https://docs.spring.io/spring-boot/docs/current/reference/html/common-application-properties.html

找到下面的属性

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

如果你使用了 H2 内部存储数据库，它里面确定了 Spring Boot 用来安装你的 H2 数据库的名字。

77、如果 H2 不在类路径里面，会出现上面情况？

将会报下面的错误

```
Cannot determine embedded database driver class for database type NONE
```

把 H2 添加至 pom.xml 中，然后重启你的服务器

```
<dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
<scope>runtime</scope>
</dependency>
```

78、你能否举一个以 ReadOnly 为事务管理的例子？

当你从数据库读取内容的时候，你想把事物中的用户描述或者是其它描述设置为只读模式，以便于 Hebernate 不需要再次检查实体的变化。这是非常高效的。

79、发布 Spring Boot 用户应用程序自定义配置的最好方法是什么？

@Value 的问题在于，您可以通过应用程序分配你配置值。更好的操作是采取集中的方法。
你可以使用 @ConfigurationProperties 定义一个配置组件。

```
@Component
@ConfigurationProperties("basic")
public class BasicConfiguration {
    private boolean value;
    private String message;
    private int number;
```

你可以在 application.properties 中配置参数。

```
basic.value: true
basic.message: Dynamic Message
basic.number: 100
```

80、配置文件的需求是什么？

企业应用程序的开发是复杂的，你需要混合的环境：

- Dev
- QA
- Stage
- Production

在每个环境中，你想要不同的应用程序配置。

> 配置文件有助于在不同的环境中进行不同的应用程序配置。

Spring 和 Spring Boot 提供了你可以制定的功能。

- 不同配置文件中，不同环境的配置是什么？
- 为一个制定的环境设置活动的配置文件。

Spring Boot 将会根据特定环境中设置的活动配置文件来选择应用程序的配置。

81、如何使用配置文件通过 Spring Boot 配置特定环境的配置？

配置文件不是设别环境的关键。

在下面的例子中，我们将会用到两个配置文件

- dev
- prod

缺省的应用程序配置在 application.properties 中。让我们来看下面的例子：

application.properties

```
basic.value= true
basic.message= Dynamic Message
basic.number= 100
```

我们想要为 dev 文件自定义 application.properties 属性。我们需要创建一个名为 application-dev.properties 的文件，并且重写我们想要自定义的属性。

application-dev.properties

```
basic.message: Dynamic Message in DEV
```

一旦你特定配置了配置文件，你需要在环境中设定一个活动的配置文件。

有多种方法可以做到这一点：

- 在 VM 参数中使用 Dspring.profiles.active=prod
- 在 application.properties 中使用 spring.profiles.active=prod

文章来源： http://www.3xmq.com/article/1522809264295

82、我们如何使用Maven设置Spring Boot应用程序？

我们可以像在任何其他库中一样在Maven项目中包含Spring Boot。但是，最好的方法是从 `spring-boot-starter-parent` 项目继承并声明依赖于Spring Boot启动器。这样做可以让我们的项目重用Spring Boot的默认设置。 继承 `spring-boot-starter-parent` 项目非常简单 - 我们只需要在 `pom.xml` 中指定一个 `parent` 元素：

```
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.1.6.RELEASE</version>
</parent>
```

我们可以在Maven 中央仓库找到最新版本的 `spring-boot-starter-parent`。 上面的方式很方便但是并不一定符合实际需要。例如公司要求所有项目依赖构建从一个标准BOM开始，我们就不能按上面的方式进行。 在这种情况下，我们可以进行如下引用：

```
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-dependencies</artifactId>
            <version>2.1.6.RELEASE</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
```

83、如何禁用特定的自动配置？

如果我们要禁用特定的自动配置，我们可以使用 `@EnableAutoConfiguration` 注解的 `exclude` 属性来指示它。如下禁用了数据源自动配置 `DataSourceAutoConfiguration` ：

```
// other annotations
@EnableAutoConfiguration(exclude = DataSourceAutoConfiguration.class)
public class MyConfiguration { }
```

如果我们使用 `@SpringBootApplication` 注解。 它具有 `@EnableAutoConfiguration` 作为元注解 - 我们同样可以配置exclude属性来禁用自动配置：

```
// other annotations
@SpringBootApplication(exclude = DataSourceAutoConfiguration.class)
public class MyConfiguration { }
```

我们还可以使用 `spring.autoconfigure.exclude` 环境属性禁用自动配置。在 `application.properties` (也可以是 `application.yml`)配置文件设置如下也可以达到同样的目的：

```
spring.autoconfigure.exclude=org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConfiguration
```

84、Spring boot支持哪些外部配置？

Spring Boot支持外部配置，允许我们在各种环境中运行相同的应用程序。我们可以使用 `properties` 文件， `YAML文件` ，环境变量，系统属性和命令行选项参数来指定配置属性。 然后，我们可以访问使用这些属性@Value注释，经由绑定对象 的 `@ConfigurationProperties` 注释或 `Environment` 环境抽象类注入。 以下是最常见的外部配置来源：

- 命令行属性：命令行选项参数是以双连字符开头的程序参数，例如 `-server.port = 8080` 。Spring Boot将所有参数转换为属性，并将它们添加到环境属性集中。

- 应用程序属性：应用程序属性是从`application.properties`文件或其`YAML`对应文件加载的属性。默认情况下，Spring Boot会在当前目录，类路径根或其`config`子目录中搜索此文件。 特定于配置文件的属性：特定于配置文件的属性从`application- {profile} .properties`文件或其`YAML`对应文件加载。`{profile}`占位符是指活性轮廓。这些文件与非特定属性文件位于相同位置，并且优先于非特定属性文件。

85、如何对Spring Boot应用进行测试？

在为Spring应用程序运行集成测试时，我们必须有一个`ApplicationContext`。 为了简化测试，Spring Boot为测试提供了一个特殊的注释`@SpringBootTest`。此批注从其`classes`属性指示的配置类创建`ApplicationContext`。 如果未设置classes属性，Spring Boot将搜索主配置类。搜索从包含测试的包开始，直到找到使用@SpringBootApplication或@SpringBootConfiguration注释的类。 请注意，如果我们使用`JUnit 4`，我们必须用`@RunWith（SpringRunner.class）`装饰测试类。

86、Spring Boot Actuator有什么用？

`Spring Boot Actuator`可以帮助你监控和管理Spring Boot应用，比如健康检查、审计、统计和HTTP追踪等。所有的这些特性可以通过`JMX`或者`HTTP endpoints`来获得。 Actuator同时还可以与外部应用监控系统整合，比如`Prometheus`, `Graphite`, `DataDog`, `Influx`, `Wavefront`, `New Relic`等。这些系统提供了非常好的仪表盘、图标、分析和告警等功能，使得你可以通过统一的接口轻松的监控和管理你的应用。`Actuator`使用`Micrometer`来整合上面提到的外部应用监控系统。这使得只要通过非常小的配置就可以集成任何应用监控系统。 将Spring Boot Actuator集成到一个项目中非常简单。我们需要做的就是在`pom.xml`文件中包含`spring-boot-starter-actuator`启动器：

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
```

`Spring Boot Actuator`可以使用`HTTP`或`JMX`端点公开操作信息。但是，大多数应用程序都使用`HTTP`，其中端点的标识和/执行器前缀形成`URL`路径。 以下是Actuator提供的一些最常见的内置端点：

- `auditevents`：公开审计事件信息
- `env`：公开环境属性
- `health`：显示应用程序运行状况信息
- `httptrace`：显示HTTP跟踪信息
- `info`：显示任意应用程序信息
- `metrics`：显示指标信息
- `mappings`：显示所有@RequestMapping路径的列表
- `scheduledtasks`：显示应用程序中的计划任务
- `threaddump`：执行线程转储
- `beans：所有加载的spring bean

更多关于`Spring Boot Actuator`的信息可查看Spring Boot 2.x 中的 Actuator 。 **请注意：生产使用Actuator务必保护好这些端点，避免未授权的访问请求。**

87、SpringBoot 中静态首页默认位置可以放在哪里？

当我们应用根目录时，可以直接映射，将 index.html 放入下面的位置：

```
classpath:/META-INF/resources/index.html
classpath:/resources/index.html
classpath:/static/index.html
classpath:/public/index.html
```

89、SpringBoot 中静态资源直接映射的优先级是怎样的？

SpringBoot 静态资源直接映射为/**，可以通过根目录来访问。/META-INF/resources/webjars/映射为/webjars/，通过访问 /webjar 访问。优先级顺序为：META-INF/resources > resources > static > public。

90、继承 WebMvcConfigurerAdapter 抽象类，常用的重写方法列举几个？

WebMvcConfigurerAdapter 实现 WebMvcConfigurer 接口，常用的可能需要重写的方法有下面几个：

```
/** 解决跨域问题 **/
public void addCorsMappings(CorsRegistry registry) ;
/** 添加拦截器 **/
void addInterceptors(InterceptorRegistry registry);
/** 这里配置视图解析器 **/
void configureViewResolvers(ViewResolverRegistry registry);
/** 配置内容裁决的一些选项 **/
void configureContentNegotiation(ContentNegotiationConfigurer configurer);
/** 视图跳转控制器 **/
void addViewControllers(ViewControllerRegistry registry);
/** 静态资源处理 **/
void addResourceHandlers(ResourceHandlerRegistry registry);
/** 默认静态资源处理器 **/
void configureDefaultServletHandling(DefaultServletHandlerConfigurer
configurer);
```

91、@SpringBootApplication 引入了哪3个重要的注解？

@SpringBootConfiguration、@EnableAutoConfiguration、@ComponentScan。其它的 4 个 @Target、@Retention、@Documented、@Inherited，也重要，但应该不是本题想问的知识点。

92、@SpringBootApplication 注解中的属性相当于哪几个注解？

等价于以默认属性使用 @Configuration，@EnableAutoConfiguration 和 @ComponentScan。