

Spring Boot 是微服务中最好的 Java 框架. 我们建议你能够成为一名 Spring Boot 的专家。本文精选了三十五个常见的 Spring Boot 知识点，祝你一臂之力！

问题一 **Spring Boot、Spring MVC 和 Spring 有什么区别？**

1、 Spring

Spring 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。

2、Spring MVC

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

3、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.InternalResourceView"
    <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。

问题二 什么是自动配置？

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/" />
```

我们能否带来更多的智能？当一个 MVC JAR 添加到应用程序中的时候，我们能否自动配置一些 beans？

Spring 查看（CLASSPATH 上可用的框架）已存在的应用程序的配置。在此基础上，Spring Boot 提供了配置应用程序和框架所需要的基本配置。这就是自动配置。

问题三 什么是 Spring Boot Starter ？

启动器是一套方便的依赖描述符，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 Spring 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。

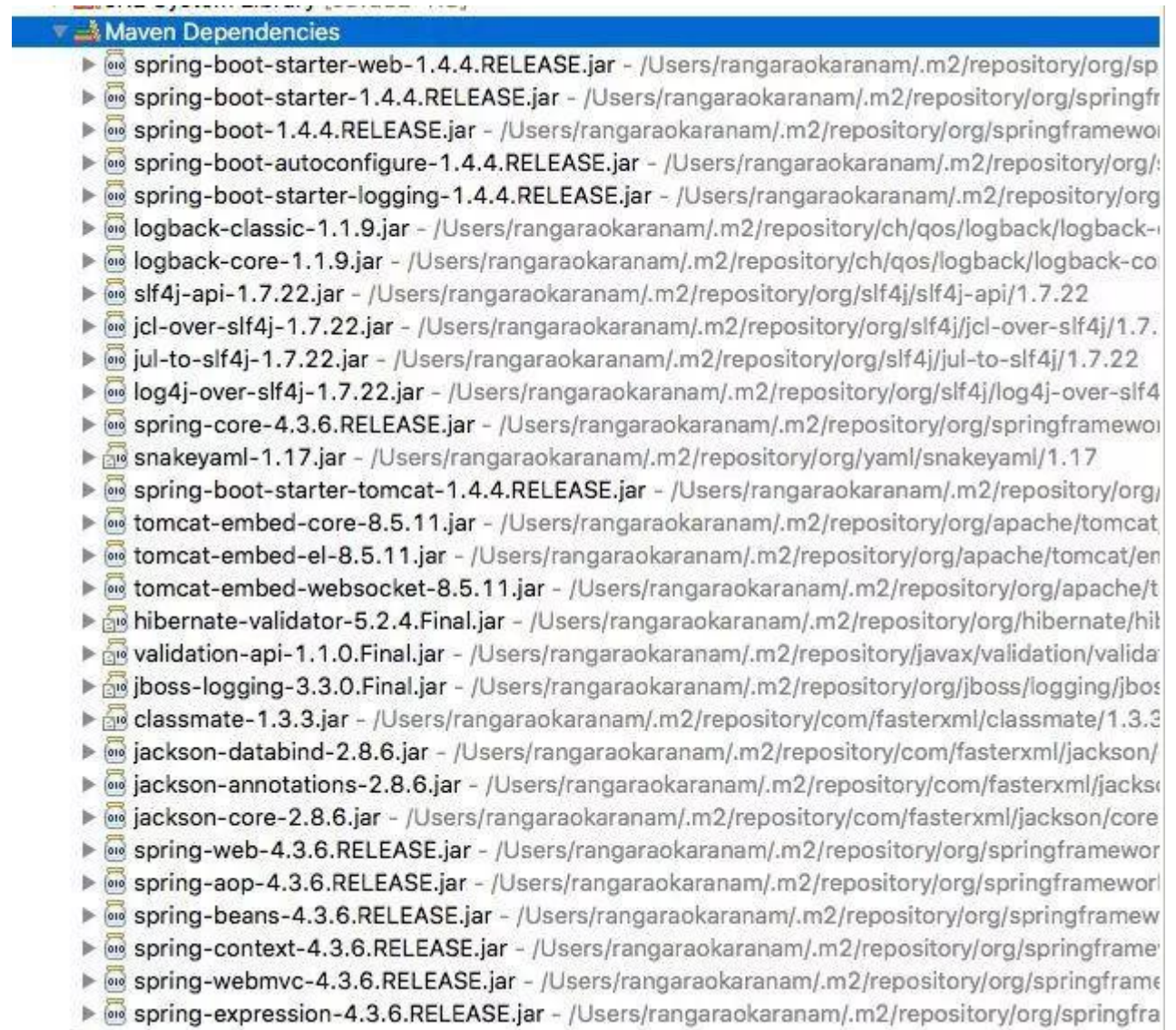
问题四 你能否举一个例子来解释更多 Staters 的内容？

让我们来思考一个 Starter 的例子 -Spring Boot Starter 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

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

Embedded 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。

问题五 **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;

问题六 **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？

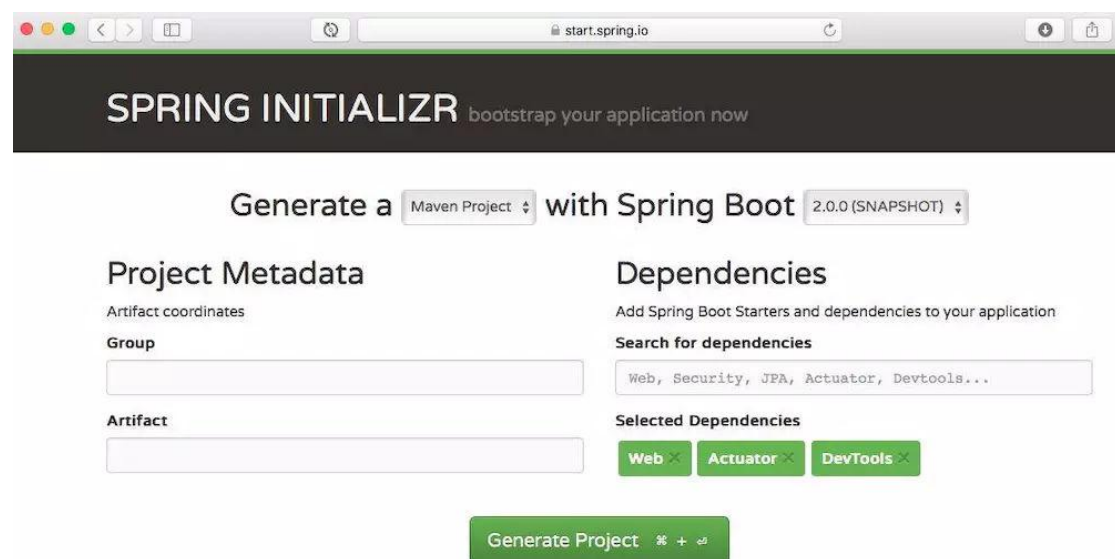
Spring Boot 2.0 需要 Java8 或者更新的版本。Java6 和 Java7 已经不再支持。

推荐阅读：

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

问题七 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。

The image shows a web browser window with the URL 'start.spring.io'. The page has a dark header with the text 'SPRING INITIALIZR bootstrap your application now'. Below the header, there's a section titled 'Generate a' followed by a dropdown menu showing 'Maven Project', then 'with Spring Boot' followed by another dropdown menu showing '2.0.0 (SNAPSHOT)'. The main content area is divided into two columns. The left column is titled 'Project Metadata' and contains 'Artifact coordinates' with 'Group' and 'Artifact' input fields. The right column is titled 'Dependencies' and contains 'Add Spring Boot Starters and dependencies to your application' with a 'Search for dependencies' input field showing 'Web, Security, JPA, Actuator, Devtools...'. Below the search field, there's a 'Selected Dependencies' section with three buttons: 'Web', 'Actuator', and 'DevTools', each with a close icon. At the bottom center, there's a large green button labeled 'Generate Project'.

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 `GenerateProject`

将项目导入 Eclipse。文件 - 导入 - 现有的 Maven 项目

问题八 Spring Initializr 是创建 Spring Boot Projects 的唯一方法吗？

不是的。

Spring Initializr 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！

问题九 为什么我们需要 **spring-boot-maven-plugin**?

spring-boot-maven-plugin 提供了一些像 **jar** 一样打包或者运行应用程序的命令。

spring-boot:run 运行你的 **SpringBooty** 应用程序。

spring-boot: repackage 重新打包你的 **jar** 包或者是 **war** 包使其可执行

spring-boot: start 和 **spring-boot: stop** 管理 **Spring Boot** 应用程序的生命周期（也可以说是为了集成测试）。

spring-boot:build-info 生成执行器可以使用的构造信息。

问题十 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。

问题十一 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**）。

问题十二 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：


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

错误: HAL browser gives me unauthorized error - Full authentication 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

问题十三 什么是 **Spring Data**?

来自: [//projects.spring.io/spring-data/](http://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。

问题十四 什么是 Spring Data REST?

Spring Data REST 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: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。

问题十五 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。

问题十六 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**。

问题十七 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>
```

问题十八 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>

问题十九 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。

问题二十 **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 **RequestMapping** 的一个延伸，目的是为了提提高清晰度。

问题二十一 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。

问题二十二 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！

问题二十三 可以配置 **application.properties 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>

问题二十四 **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。

问题二十五 业务边界应该从哪一层开始？

我们建议在服务层管理义务。商业业务逻辑在商业层或者服务层，与此同时，你想要执行的业务管理也在该层。

问题二十六 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。

问题二十七 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-starter-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。

问题二十八 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>

问题二十九 我们如何连接一个像 **MySQL** 或者 **Oracle** 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！

问题三十 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

Name of the datasource.

spring.datasource.name=testdb

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。

问题三十一 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>
```

问题三十二 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。

问题三十三

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`

问题三十四 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。

问题三十五 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 `spring.profiles.active=prod`

在 `application.properties` 中使用 `spring.profiles.active=prod`

为什么某些人会一直比你优秀，是因为他本身就很优秀还一直在持续努力变得更优秀，而你是不是还在满足于现状内心在窃喜！