基础知识篇

Q1. 什么是 Spring Boot 并说明它的必要性？

Spring Boot 是 Spring 的一个模块，目的是要简化 Java 开发中 Spring 框架的使用。它可以在 Spring 的基础上创建独立的、可直接运行的应用程序。因此，它删除了许多配置和依赖项。针对快速的应用程序开发，Spring Boot 框架提供了自动依赖解决方案、嵌入式 HTTP 服务器、自动配置、管理端点和 Spring Boot CLI。 由此可见，Spring Boot 不仅提高了工作效率，而且为编写自己的业务逻辑提供了很多便利。

Q2. Spring Boot 和 Spring 的区别是什么？

Spring	Spring Boot
基于 Java Web 的应用程序框架	Spring 模块
提供可自定义的 Web 应用程序工具和库	用于创建一个可执行的 Spring 应用程序项目
Spring 比 Spring Boot 更加复杂	Spring Boot 没有 Spring 框架复杂
采取一种无偏见的观点	对平台有自己的看法

Q3. Spring Boot 有什么优点？

Spring Boot 的优点如下：

- 提供自动加载一组默认配置的功能来快速启动应用程序
- 闯将对大型项目类中常见的一些列肺功能特性的独立应用程序
- 内置 Tomcat、Servlet 容器和 Jetty，避免使用 WAR 文件
- Spring Boot 提供了视图功能，减少了开发人员的工作量，简化了 Maven 配置
- 提供 CLI 工具来开发和测试应用程序
- 附带 Spring 启动器，以确保依赖项管理，并提供各种安全指标
- 包含广泛的 API，用于监控和管理开发和 prod 中的应用程序。
- 与 Spring JDBC、Spring ORM、Spring Data、Spring Security 等 Spring 生态系统集成，避免样板代码。

Q4. Srping Boot 的特性

以下是 Spring Boot 的几个重要特性：

1. **Spring CLI**：允许使用 Groovy 编写 Spring 引导应用程序，并避免了样板代码。
2. **启动依赖项**：在此功能的帮助下，Spring Boot 将公共依赖项聚集在一起，最终提高了工作效率。
3. **Spring Initializer**：这基本上是一个 Web 应用程序，它可以自动创建一个内部项目结构，不必再手动设置。
4. **自动配置**：Spring Boot 的自动配置特性可以为正在处理的项目加载默认配置。这正方式可以避免不必要的 WAR 文件。
5. **Spring 驱动器**：这一功能为运行 Spring Boot 应用程序提供帮助。
6. **日志和安全性**：Spring Boot 的日志和安全性特性，确保所有使用 Spring Boot 的应用程序都得到了适当的保护，没有任何麻烦。

Q5. 什么是 Spring Boot，什么是可用的启动器？

Spring 启动程序是一组方便的依赖项管理提供程序，可以在应用程序中使用它们来启用依赖项。这些启动器使开发变得简单和快速。所有可用的启动程序都在 org.springframework 下。引导组。以下是一些流行的开胃菜：

- spring-boot-starter：这是核心 starter，包括日志记录、自动配置支持和 YAML。
- spring-boot-starter-jdbc：该启动程序用于与 JDBC 的 HikariCP 连接池
- spring-boot-starter-web：是使用 Spring MVC 构建 Web 应用程序（包括 RESTful 应用程序）的起点
- spring-boot-starter-Dat-JPA：是在 Hibernate 中使用 Spring Data JPA 的第一步
- spring-boot-starter-Security：是用于 Spring Security 的启动器
- spring-boot-starter-AOP：这个 starter 用于使用 AspectJ 和 Spring AOP 进行面向方面的编程
- spring-boot -starter-test：用于测试 Spring 引导应用程序

Q6. 什么是 Spring Boot 依赖项管理？

Spring Boot 依赖项管理主要用于自动管理依赖项和配置，而不需要为任何依赖项指定版本。

Q7. 提到 JPA 和 Hibernate 之间的区别

@RequestMapping	@RestController
JPA 是一个数据访问抽象，用于减少样板代码的数量	Hibernate 是 Java Persistence API 的一个实现，提供了松散耦合

Q8. RequestMapping 和 GetMapping 的区别是什么？

@GetMapping 是一个复合注释，它充当 @RequestMapping(method = RequestMethod.GET) 的快捷方式。这两种方法都支持。consume 选项有：

```
consume = " text / plain
```

```
consume = {" text/plain ",  " application/* "}
```

Q9. Spring Boot 的核心注解是哪个？它主要由哪几个注解组成的？

@SpringBootApplication 是启动类上面的注解，同时也是 Spring Boot 的核心注解，它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：打开或关闭自动配置，如关闭数据源自动配置功能：`@SpringBootApplication(exclude = { DataSourceAutoConfiguration.class })`。
- @ComponentScan：Spring 组件扫描。

Q10. 由 Spring Boot 配置的默认 H2 数据库的名称是什么？

默认的 H2 数据库的名称是 testdb。请参考下面：

```
spring.datasource.name=testdb    #数据源的名称
```

注意：为以防万一，您可以使用 Spring Boot 的名称来命名正在使用的 H2 内存数据库。

Q11. 解释 Spring Actuator 及其优点

Spring Actuator 是 Spring Boot 的一个很酷的特性，通过它可以看到正在运行的应用程序中发生了什么。每当你想调试你的应用程序，并需要分析日志时你需要了解在应用程序中发生了什么。在这种情况下，Spring Actuator 提供了对诸如标识 Bean、CPU 使用情况等特性的简单访问。Spring Actuator 提供了一种非常简单的方法来访问准备生产的 REST 点并从 Web 获取各种信息。使用 Spring Security 的内容协商策略保护这些点。

Q12. @SpringBootApplication 和 @EnableAutoConfiguration 注释有什么区别？

@SpringBootApplication	@EnableAutoConfiguration
在主类或引导类中使用	用于在项目中启用自动配置和组件扫描
它是 @Configuration、@ComponentScan 和 @EnableAutoConfiguration 注释的组合。	它是 @Configuration 和 @ComponentScan 注释的组合

应用实战篇

Q13. 解释如何使用 Maven 创建 Spring 启动应用程序

使用 Maven 来创建 Spring Boot 应用程序的方法有很多，下面列出其中的一部分：

- Spring Boot CLI
- Spring Starter 项目向导
- Spring Initializr
- Spring Maven 项目

Q14. 说明外部配置的可能来源

Spring Boot 通过提供外部配置支持可以允许开发人员在不同的环境中运行相同的应用程序。它通过使用环境变量、属性文件、命令行参数、YAML 文件和系统属性来说明所需的配置属性。另外，@value 注释可以获得对属性的访问。由此可见，外部配置最可能的来源如下：

- 应用程序属性：默认情况下，Spring Boot 在当前目录、类路径、根目录或配置目录中搜索应用程序属性文件或其 YAML 文件以加载属性。
- 命令行属性：Spring Boot 提供命令行参数，并将这些参数转换为属性，然后将它们添加到环境属性集。
- 特定于概要文件的属性：这些属性是从应用程序中加载的 -{概要文件}。属性文件或其 YAML 文件。此文件与非特定属性文件位于相同的位置，{profile} 占位符指的是活动的概要文件。

Q15. 您能解释一下当 Spring 启动应用程序“作为 Java 应用程序运行”时在后台会发生什么吗？

当 Spring 启动应用程序以“作为 Java 应用程序运行”的方式执行时，当它看到您正在开发 Web 应用程序时，就会自动启动 Tomcat 服务器。

当 Spring Boot 应用程序以 Java 应用程序”的方式执行的时候，就会启动 Tomcat 服务器。

Q16. Spring Boot 系统的最低要求是什么？

Spring boot 2.1.7.RELEASE 需要

- Java 8 +
- Spring Framework 5.1.9 +

明确的支持

- Maven 3.3 +
- Gradle 4.4 +

Servlet 容器支持

- Tomcat 9.0 - Servlet 版本 4.0
- Jetty 9.4 - Servlet 版本 3.1
- Undertow 2.0 - Servlet 版本 4.0

Q17. 解释什么是 thymeleaf 以及如何使用 thymeleaf？

Thymeleaf 是一个服务器端 Java 模板引擎用于 Web 应用程序。它的目标是为您的 Web 应用程序带来自然的模板，并且能够很好地与 Spring 框架和 HTML5 Java Web 应用程序集成。要使用 Thymeleaf，您需要在 pom.xml 文件中添加以下代码：

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

Q18. 我们可以在 Spring Boot 中更改嵌入式 Tomcat 服务器的端口吗？

是的，我们可以使用应用程序属性文件更改嵌入式 Tomcat 服务器的端口。在这个文件中，必须添加“server”属性。并将其分配到您希望的任何端口。例如，如果希望将其分配给 8081，则必须提到 *server.port=8081*。一旦提到端口号，应用程序属性文件将在 Spring 引导时自动加载，所需的配置将应用于应用程序。

Q19. 如何在 Spring Boot 中启用 HTTP/2 支持

您可以在 Spring Boot by 中启用 HTTP/2 支持：

```
server.http2.enabled=true
```

Q20. 如何在 Spring Boot 执行器中创建自定义端点？

在 Spring Boot 2.x 中创建自定义端点可以使用 @Endpoint 注释。Spring Boot 还在 Spring MVC、Jersey 等的帮助下，通过 HTTP 使用 @WebEndpointor、@WebEndpointExtension 来公开端点。

Q21. 将 Spring 引导 Web 应用程序部署为 JAR 和 WAR 文件的步骤是什么？

要部署 Spring Boot Web 应用程序，只需在 pom.xml 文件中添加以下插件：

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

通过使用上面的插件，您将得到一个执行包阶段的 JAR。这个 JAR 将包含所有必需的库和依赖项。它还包含一个嵌入式服务器。因此，您基本上可以像普通 JAR 文件一样运行应用程序。

注意：构建一个 JAR 文件，需要在 pom.xml 中设置：

```
<packaging>jar</packaging>
```

类似地，如果您想要构建一个 WAR 文件，那么需要在 pom.xml 中设置：

```
<packaging>war</packaging>
```

Q22. 您能解释一下如何使用 Spring Boot 部署到不同的服务器吗？

要使用 Spring Boot 部署不同的服务器，请遵循以下步骤：

- 从项目中生成 WAR 包
- 然后，将 WAR 文件部署到您喜欢的服务器上

注意：在服务器上部署 WAR 文件的步骤取决于您选择的服务器。

Q23. 在 Spring Boot 中添加自定义 JS 代码的步骤是什么？

使用 Spring Boot 添加自定义 JS 代码的步骤如下：

- 在 resources 文件夹下创建一个文件夹，并将其命名为 static
- 可以将静态内容放入该文件夹

注意：以防浏览器抛出未经授权的错误，您可以调整安全策略或在日志文件中搜索密码，并放在在请求头中。

Q24. 使用 Spring Boot 公开自定义应用程序配置的最佳方式是什么？

在 Spring Boot 中公开自定义应用程序配置的一种方法是使用 @Value 注释。但是，这个注释的惟一问题是所有的配置值都分布在整个应用程序中。相反，您可以使用集中式方法。

通过集中式方法，我的意思是您可以使用 @ConfigurationProperties 定义配置组件，如下所示：

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

根据上面的代码片段，应用程序中配置的值。属性如下：

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

Q25. 连接外部数据库（如 MySQL 或 Oracle ）的步骤是什么？

要连接外部数据库，您必须遵循以下步骤：

- 首先，将 MySQL 连接器的依赖项添加到 pom.xml
- 然后，从 pom.xml 中删除 H2 依赖项
- 第三，设置 MySQL 数据库并配置到 MySQL 数据库的连接
- 最后，重启项目

理论实现篇

Q26. Spring Boot DevTools 需要什么？

Spring Boot 开发工具是一组精心设计的工具，旨在简化应用程序的开发过程。如果应用程序在生产环境中运行，则此模块将自动禁用，缺省情况下也不包括重新打包存档。因此，Spring Boot 开发人员工具将属性应用于相应的开发环境要包含 DevTools，您只需将以下依赖项添加到 pom.xml 文件中：

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

Q27. 介绍使用 Spring Initializr 创建 Spring 启动项目的步骤

Spring Initializr 是 Spring 提供的一个 Web 工具。在这个工具的帮助下，您可以通过提供项目细节来创建 Spring Boot 项目。使用 Spring Initializr 创建 Spring 启动项目需要遵循以下步骤：

- 选择 Maven 项目和所需的依赖项。然后填写其他需要的细节，如组、工件，然后单击 Generate Project。
- 下载项目后，将项目解压缩到您的系统中。
- 接下来，您必须使用 Spring 工具套件 IDE 上的 import 选项导入这个项目。

在导入项目时，请记住，您必须选择项目类型为 Maven，源项目应该包含 pom.xml 文件。

一旦执行了上述所有步骤，您将看到 Spring Boot 项目是使用所有必需的依赖项创建的。

Q28. 介绍使用 JDBC 将 Spring 启动应用程序连接到数据库的步骤

Spring Boot starter 项目提供了将应用程序与 JDBC 连接所需的库。因此，例如，如果你只需要创建一个应用程序，并将其与 MySQL 数据库连接，你可以遵循以下步骤。

STEP 1：创建 MySQL 数据库：

```
CREATE DATABASE spring_boot;
```

STEP 2：然后必须在这个数据库中创建一个表。

STEP 3：现在，创建一个 Spring Boot 项目并提供所需的详细信息

STEP 4：添加 JDBC、MySQL 和 Web 依赖项。

STEP 5：创建项目之后，必须将数据库配置为应用程序属性：

```
spring.datasource.url=jdbc:mysql://localhost:3306/spring_boot
spring.datasource.username=root
spring.datasource.password=springboot
spring.jpa.hibernate.ddl-auto=create-drop
```

STEP 6：Java 类应该有以下代码：

```java
package com.springboot;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class SampleApplication {
    public static void main(String[] args) {
        SpringApplication.run(SampleApplication.class, args);
    }
}
```

STEP 7：接下来，你必须创建一个控制器来处理 HTTP 请求，通过以下代码：

```java
package com.springboot;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.web.bind.annotation.RestController;
@RestController
public class JdbcController {
    @Autowired
    JdbcTemplate jdbc;
    @RequestMapping("/insert")
    public String index(){
        jdbc.execute("insert into customers(name)values('Aryya')");
        return "Data Entry Successful";
    }
}
```

STEP 8：最后，将此项目作为 Java 应用程序执行。

STEP 9：接下来，打开 URL（localhost:8080/insert），您将看到数据输入成功的输出。您还可以进一步检查是否将数据输入到表中。

Q29. 在 Spring Boot 中 @RequestMapping 和 @RestController 注释的用途是什么？

@RequestMapping	@RestController
该注释用于提供路由信息，并告诉 Spring 任何 HTTP 请求都必须映射到相应的方法。	此注释用于将 @ResponseBody 和 @Controller 注释添加到类中
要使用这个注释，您必须导入	要使用这个注释，您必须导入

@RequestMapping	@RestController
org.springframework.web.bind.annotation.RequestMapping;	org.springframework.web.bind.annotation.RestController;

示例：假设您有一个方法 example()，它应该映射 /example 的 URL。

```
package com.springboot;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
@RestController
public class SampleController {
    @RequestMapping("/example")
    public String example(){
        return"Welcome To SpringBoot";
    }
}
```

Q30. 什么是 Spring Boot CLI，如何使用 Boot CLI 执行 Spring Boot 项目？

Spring Boot CLI 是官方支持 Spring 框的工具。执行 Spring Boot 项目的步骤如下：

- 从官方网站下载 CLI 工具并解压，Spring 启动程序在其设置中的 bin 文件夹内。
- 由于 Spring Boot CLI 执行 Groovy 文件，因此需要为 Spring Boot 应用程序创建一个 Groovy 文件。为此，打开 terminal 并进入 bin 文件夹目录，打开一个 Groovy 文件（例如 Sample.groovy），在这个文件中创建一个控制器如下：

```
@RestController
public class Sample {
    @RequestMapping("/example")
    String index(){
        <h1>"Welcome To SpringBoot"</h1>;
    }
}
```

然后通过以下方式执行 Groovy 文件：

```
./spring run Sample.groovy;
```

一旦项目被执行，进入 URL（localhost:8080:/example），你就会看到结果。

Q31. Spring 解释数据

Spring Data 的目标是使开发人员能够轻松地使用关系和非关系数据库、基于云的数据服务和其他数据访问技术。因此，基本上，它使数据访问变得容易，并且仍然保留底层数据。

Q32. 您对 Spring Boot 中的自动配置有什么理解？如何禁用自动配置？

Auto-configuration 用于自动配置应用程序所需的配置。例如，如果在应用程序的类路径中有一个数据源 Bean，那么它会自动配置 JDBC 模板。在自动配置的帮助下，您可以轻松地创建 Java 应用程序，因为它可以自动配置所需的 Bean、控制器等。

要禁用自动配置属性，您必须排除 @EnableAutoConfiguration 属性，在不希望应用该属性的场景中。

```
@EnableAutoConfiguration(exclude={DataSourceAutoConfiguration.class}
```

如果类不在类路径中，那么要排除自动配置，你必须提到以下代码：

```
@EnableAutoConfiguration(excludeName={Sample.class})
```

除此之外，Spring Boot 还提供了通过使用 Spring .autoconfigure.exclude 属性来排除自动配置类列表的功能。您可以继续，并将其添加到应用程序中，属性或添加以逗号分隔的多个类。

Q33. 你能举一个事务管理中 ReadOnly 为 true 的例子吗？

事务管理中 ReadOnly 为 true 的例子如下：

假设您必须从数据库中读取数据。例如，假设您有一个客户数据库，希望读取客户详细信息，如 customerID 和 customername。如果不想检查实体中的更改，需要将事务设置为只读。

Q34. 我们可以在 Spring Boot 中创建一个非 Web 应用程序吗？

是的，我们可以通过从类路径中删除 Web 依赖项以及改变 Spring Boot 创建应用程序上下文的方式来创建非 Web 应用程序。

Q35. 叙述 YAML 文件比属性文件的优优势在哪里，以及在 Spring Boot 中加载 YAML 文件的不同方式

YAML 文件比属性文件的优势在于数据以分层格式存储。如果出现问题，开发人员很容易进行调试。当您在类路径上使用 SnakeYAML 库时，SpringApplication 类支持 YAML 文件作为属性的替代。在 Spring Boot 中加载 YAML 文件的不同方法如下：

- 使用 YamlMapFactoryBean 加载 YAML 作为地图
- 使用 YamlPropertiesFactoryBean 加载 YAML 作为属性

Q36. 如何在不进行任何配置的情况下选择 Hibernate 作为 JPA 的默认实现？

当我们使用 Spring Boot 默认配置时，Spring-Boot-starter-data-jpa 依赖项会自动添加到 pom.xml 文件中。现在，由于这个依赖项对 JPA 和 Hibernate 有一个可传递的依赖项，Spring Boot 会自动将 Hibernate 配置为 JPA 的默认实现，只要它在类路径中看到 Hibernate。

Q37. 您对 Spring Date REST 的理解是什么？

Spring Data REST 用于公开 Spring 数据库周围的 RESTful 资源。例如：

```
@RepositoryRestResource(collectionResourceRel = "sample", path = "sample")
public interface SampleRepository
extends CustomerRepository<sample, Long> {
```

现在，为了公开 REST 服务，您可以 POST 一下的信息：

```
{"customername": "Rohit"}
```

响应内容：

```
{
    "customername": "Rohit"
    "_links": {
        "self": {
            "href": "http://localhost:8080/sample/1"
        },
        "sample": {
            "href": "http://localhost:8080/sample/1"
        }
    }
}
```

注意，响应内容包含新创建的资源的 href。

Q38. 事务的边界应该从哪一层开始？

事务的边界应该从服务层开始，因为业务的事务逻辑存在于该层本身。

Q39. path = "sample"， collectionResourceRel = "sample" 如何使用 Spring Data Rest？

```
@RepositoryRestResource(collectionResourceRel = "sample", path = "sample")public interface
SampleRepository extendsPagingAndSortingRepository<Sample, Long>
```

- path：用来配置资源导出的路径。
- collectionResourceRel：此值用于生成集合资源的链接。

Q40. 如何配置 Log4j 进行日志记录？

Spring Boot 支持 Log4j2 配置，排除 Logback 并包含 Log4j2 来为日志做记录。但这种方式只有在使用 starter 项目时才可以。

Q41. 提到 WAR 和嵌入式容器之间的区别

WAR	嵌入式容器
WAR 在很大程度上得益于 Spring Boot	Spring Boot 只有一个组件，在改进期间使用

Q42. 你认为对个人资料的需求是什么？

配置文件用于提供一种方法来隔离应用程序配置的不同部分，并使其可用于各种环境。任何 @Component 或 @Configuration 都可以标记为 @Profile 来限制它的加载。假设你有多个环境

- Dev
- QA
- Stage
- Production

您希望在每个环境中都有不同的应用程序配置，您可以使用配置文件为不同的环境提供不同的应用程序配置。Spring 和 Spring Boot 提供了一些特性，您可以通过这些特性指定：

- 特定环境的活动概要
- 各种配置文件的各种环境配置。

Q43. 当 Bean 存在时，如何指示自动配置退出？

要指示一个自动配置类在 Bean 存在时退出，您必须使用 @ConditionalOnMissingBean 注释。该注释的属性如下：

- value：该属性存储要检查的 Bean 的类型
- name：该属性存储要检查的 Bean 的名称

Q44. 为什么在实际应用程序中不推荐使用 Spring Data REST？

在实际应用程序中不建议使用 Spring Data REST，在数据库实体直接公开为 REST 服务，设计 RESTful 服务时，我们考虑的两个最重要的事情是域模型和使用者。但是，在使用 Spring Data REST 时，这些参数都没有考虑，实体是直接暴露的。因此可以使用 Spring Data REST 来进行项目的初始演化。

Q45. 如果 H2 不在类路径中，会出现什么错误？

如果类路径中没有 H2，则会出现无法为数据库类型 NONE 确定嵌入式数据库驱动程序类的情况。

要解决此错误，请将 H2 的依赖添加到 pom.xml 文件中，然后重新启动服务器。

依赖内容如下：

```
@RepositoryRestResource(collectionResourceRel = "sample", path = "sample")
public interface SampleRepository extends
PagingAndSortingRepository<Sample, Long>
```

Q46. 使用 profiles 配置特定环境的方式是什么？

Profiles 是识别环境的关键，这是众所周知的事实，让我们考虑以下两个概要文件的例子：

- dev
- prod

考虑应用程序属性文件中存在的以下属性：

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

如果您想要自定义应用程序属性，然后创建一个名为 application-dev 的文件，其中属性为要自定义的属性。你可以提到以下代码：

`example.message: Dynamic Message in Dev`

类似地，如果希望自定义应用程序。prod 配置文件的属性，然后您可以提到以下代码片段：

`example.message: Dynamic Message in Prod`

完成特定概要文件的配置后，必须在环境中设置活动概要文件。要做到这一点，你可以

- 设置参数 *-Dspring.profiles.active = prod*
- 在程序里设置属性文件 *spring.profiles.active=prod*

Q47. Spring Boot 可以通过依赖项启动 JPA 应用程序并连接到内存数据库 H2 吗？

可以，方法如下：

- Web starter

- h2
- data JPA starter

要包含依赖项，请参考以下代码：

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

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

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

Q48. 您对 Spring Boot 支持轻松绑定的理解是什么？

轻松绑定，是一种属性名称不需要与环境属性的键匹配的方式。在 Spring Boot，轻松绑定适用于配置属性的类型安全绑定。例如，如果使用了带有 @ConfigurationPropertie 注释的 Bean 类中的属性 sampleProp，那么它可以绑定到以下任何环境属性：

- sampleProp
- sample-Prop
- sample_Prop
- SAMPLE_PROP

Q49. 指定的数据库连接信息在哪里？它如何自动连接到 H2？

这个问题的答案很简单。正是由于 Spring Boot 自动配置配置了应用程序的依赖项。因此，数据库连接信息和数据库到 H2 的自动连接是通过 auto-configuration （自动配置）属性完成的。

Q50. 您认为在 spring-boot-starter-web 中可以使用 Jetty 而不是 Tomcat 吗？

是的，我们可以在 spring-boot-starter-web 中使用 Jetty 而不是 Tomcat，方法是删除现有的依赖项并添加以下内容：

```
<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>
```