Blog Site: Java Guides **Author:** Ramesh Fadatare

Top 25 Spring Boot Interview Questions and Answers

This book describes important Spring boot interview questions and answers. Spring Boot aims to simplify Java development with Spring by removing major pain points with respect to configuration, dependency management and ease of development.

List of Spring Boot Interview Questions and Answers

24. How to Implement Security for Spring Boot Application ?
25. How to Configure Datasource Using Spring Boot?

1. What is Spring Boot?

Spring Boot is a project built on the top of the Spring framework. It provides a simpler and faster way to set up, configure, and run both standalone and web-based applications. Spring Boot takes an opinionated view of the Spring platform and third-party libraries so we can get started with minimum fuss. Most Spring Boot applications need very little Spring configuration.
Spring Boot automatically configures required classes depending on the libraries on its classpath. Suppose your application wants to interact with DB, if there are Spring Data libraries on classpath then it automatically sets up a connection to DB along with the Data Source class.
Read more about Spring Boot at **What is Spring Boot Explained?**

2. Spring Boot Features

1. Create stand-alone Spring applications
2. Embed Tomcat, Jetty or Undertow directly (no need to deploy WAR files)
3. Provide opinionated 'starter' dependencies to simplify your build configuration
4. Automatically configure Spring and 3rd party libraries whenever possible
5. Provide production-ready features such as metrics, health checks, and externalized configuration
6. Absolutely no code generation and no requirement for XML configuration

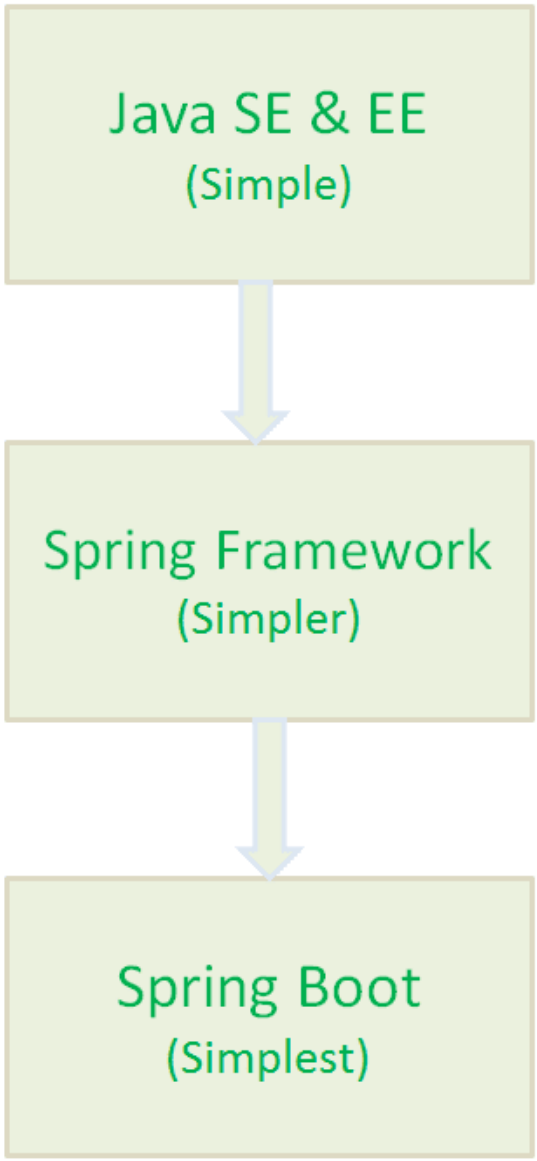
3. What are the Advantages of using Spring Boot?

1.It is very easy to develop Spring Based applications with Java.
2.It reduces lots of development time and increases productivity.
3.It avoids writing lots of boilerplate Code, Annotations and XML Configuration.
4.It is very easy to integrate Spring Boot Application with its Spring Ecosystem like Spring JDBC, Spring ORM, Spring Data, Spring Security etc.

5.It follows “Opinionated Defaults Configuration” Approach to reducing Developer effort
6.It provides Embedded HTTP servers like Tomcat, Jetty etc. to develop and test our web applications very easily.
7.It provides CLI (Command Line Interface) tool to develop and test Spring Boot(Java or Groovy) Applications from command prompt very easily and quickly.
8.It provides lots of plugins to develop and test Spring Boot Applications very easily using Build Tools like Maven and Gradle
9.It provides lots of plugins to work with embedded and in-memory Databases very easily.

4. Why We Need Spring Boot?

Spring Framework aims to simplify **J2EE Enterprise application development**. Spring Boot Framework aims to simplify Spring Development.



5. What are Spring Boot Starters?

Starters are a set of convenient dependency descriptors that you can include in your application. You get a one-stop-shop for all the Spring and related technology that you need, without having to hunt through sample code and copy paste loads of dependency descriptors.

For example, while developing the REST service or web application; we can use libraries like Spring MVC, Tomcat and Jackson – a lot of dependencies for a single application. **spring-boot-starter-web** starter can help to reduce the number of manually added dependencies just by adding **spring-boot-starter-web** dependency. So instead of manually specifying

the dependencies just add one **spring-boot-starter-web** starter as in the following example:

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

Read more in-details about Spring Boot Starters at **http://www.javaguides.net/2018/09/important-spring-boot-starters-with-examples.html**

6. How does Spring Enable Creating Production Ready Applications in Quick Time?

Spring Boot aims to enable production ready applications in quick time. Spring Boot provides a few non-functional features out of the box like caching, logging, monitoring, and embedded servers.

1.**spring-boot-starter-actuator** - To use advanced features like monitoring & tracing to your application out of the box
2.**spring-boot-starter-undertow, spring-boot-starter-jetty, spring-boot-starter-tomcat** - To pick your specific choice of Embedded Servlet Container
3.**spring-boot-starter-logging** - For Logging using logback
4.**spring-boot-starter-cache** - Enabling Spring Framework's caching support

7. What is Auto-Configuration in Spring Boot?

The problem with Spring and Spring MVC is the amount of configuration that is needed. When we use Spring MVC, we need to configure a component scan, the dispatcher servlet, a view resolver, web JARs (for delivering static content), among other things and when we use Hibernate/JPA, we would need to configure a datasource, an entity manager factory, a transaction manager, among a host of other things.
The Spring Boot auto-configuration feature tries to automatically configure your Spring application based upon the JAR dependency you have added in the classpath.

For example, if HSQLDB is present on your classpath and you have not configured any database manually, Spring will auto-configure an in-memory database for you.
By default, this auto-configuration feature is not enabled and you need to opt-in for it by adding the **@EnableAutoConfiguration** or **@SpringBootApplicaiton** annotations to one of your @Configuration classes, generally, the **Main** class which is used to run your application
Read more at **https://dzone.com/articles/what-is-spring-boot-auto-configuration**

8. What Is the Minimum Baseline Java Version for Spring Boot 2 and Spring 5?

Spring Boot 2.0 requires Java 8 or later. Java 6 and 7 are no longer supported. It also requires Spring Framework 5.0.
Recommended Reading - **https://github.com/spring-projects/spring-boot/wiki/Spring-Boot-2.0.0-M1-Release-Notes**.

9. What are Different Ways of Running Spring Boot Application?

Spring boot offers several ways of running Spring boot applications. I would like to suggest five ways we can run Spring Boot Application

1.Running from an IDE
2.Running as a Packaged Application
3.Using the Maven Plugin
4.Using External Tomcat
5.Using the Gradle Plugin

Read more at **http://www.javaguides.net/2018/09/different-ways-of-running-spring-boot-appilcation.html**

10. Name all Spring Boot Annotations?

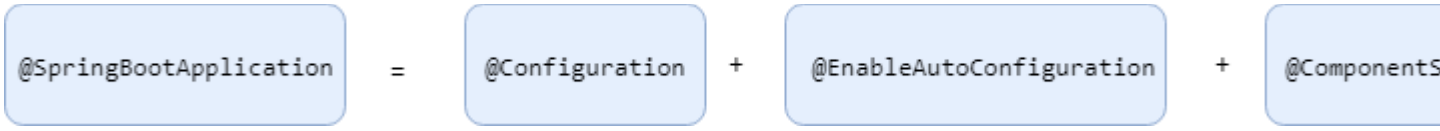
Spring Boot Annotations

1. @SpringBootApplication
2. @EnableAutoConfiguration
3. @ConditionalOnClass and @ConditionalOnMissingClass
4. @ConditionalOnBean and @ConditionalOnMissingBean
5. @ConditionalOnProperty
6. @ConditionalOnResource
7. @ConditionalOnWebApplication and @ConditionalOnNotWebApplication
8. @ConditionalExpression
9. @Conditional

Read more about each Spring Boot annotation
at **http://www.javaguides.net/2018/10/spring-boot-annotations.html**

11. What Is the Difference Between @SpringBootApplication and @EnableAutoConfiguration Annotation?

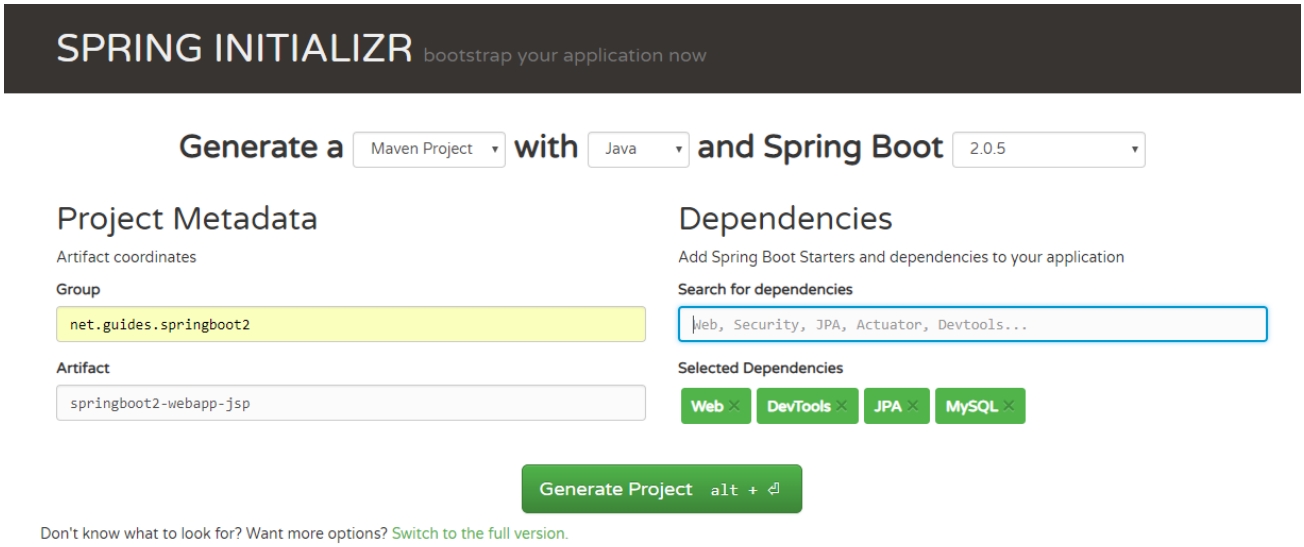
@EnableAutoConfiguration is to enable automatic configuration feature of Spring Boot application which automatically configures things if certain classes are present in Classpath. For example, it can configure *Thymeleaf*, *TemplateResolver*, and *ViewResolver* if **Thymeleaf** is present in the classpath.
@EnableAutoConfiguration also combines **@Configuration** and **@ComponentScan** annotations to enable Java-based configuration and component scanning in your project
On the other hand, **@SpringBootApplication** annotation indicates a configuration class that declares one or more **@Bean** methods and also triggers auto-configuration and component scanning. This is a convenience annotation that is equivalent to declaring **@Configuration**, **@EnableAutoConfiguration**, and **@ComponentScan**.



Read more about **@EnableAutoConfiguration** annotation with example at **Spring Boot @EnableAutoConfiguration Annotation with Example**
Read more about **@SpringBootApplication** annotation with an example at **Spring Boot @SpringBootApplication Annotation with Example**

12. What is the Easiest Approach to Create a Spring Boot Project?

Spring Initializr **http://start.spring.io/** is a great tool to bootstrap your Spring Boot projects.
Let's see how to use Spring Initializr to create Spring MVC web application.



Look at the above diagram, we have specified the following details:

- **Generate**: Maven Project
- **Java Version**: 1.8 (Default)
- **Spring Boot**:2.0.4
- **Group**: net.guides.springboot2
- **Artifact**: springboot2-webapp-jsp
- **Name**: springboot2-webapp-jsp
- **Package Name** : net.guides.springboot2.springboot2webappjsp
- **Packaging**: jar (This is the default value)
- **Dependencies**: Web, JPA, MySQL, DevTools

Once, all the details are entered, click on Generate Project button will generate a spring boot project and downloads it. Next, Unzip the downloaded zip file and import it into your favorite IDE.
Check out **Spring MVC + Spring Boot2 + JSP + JPA + Hibernate 5 + MySQL Example**

13. Why Do We Need spring-boot-maven-plugin?

The Spring Boot Maven plugin provides many convenient features:

- It collects all the jars on the classpath and builds a single, runnable "über-jar", which makes it more convenient to execute and transport your service.
- It searches for the public static void `main()` method to flag as a runnable class.
- It provides a built-in dependency resolver that sets the version number to match Spring Boot dependencies. You can override any version you wish, but it will default to Boot's chosen set of versions.

The Spring Boot Plugin has the following goals.

- `spring-boot:run` runs your Spring Boot application.
- `spring-boot:repackage` repackages your jar/war to be executable.
- `spring-boot:start` and `spring-boot:stop` to manage the lifecycle of your Spring Boot application (i.e. for integration tests).
- `spring-boot:build-info` generates build information that can be used by the Actuator.

Read more about Spring Boot Plugin at **https://docs.spring.io/spring-boot/docs/current/maven-plugin**

14. What is the Spring Boot Actuator and its Features?

Spring Boot Actuator includes a number of additional features to help you monitor and manage your application when it's pushed to production. You can choose to manage and monitor your application using HTTP or JMX endpoints. Auditing, health and metrics gathering can be automatically applied to your application.

Enabling the Actuator

The simplest way to enable the features is to add a dependency to the **spring-boot-starter-actuator** 'Starter'. To add the actuator to a Maven-based project, add the following 'Starter' dependency:

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

For Gradle, use the following declaration:

```
dependencies {
 compile("org.springframework.boot:spring-boot-starter-actuator")
}
```

Features

Endpoints: Actuator endpoints allow you to monitor and interact with your application. Spring Boot includes a number of built-in endpoints and you can also add your own. For example, the health endpoint provides basic application health information. Run up a basic application and look at `/actuator/health`.
Metrics: Spring Boot Actuator provides dimensional metrics by integrating with Micrometer.
Audit: Spring Boot Actuator has a flexible audit framework that will publish events to an `AuditEventRepository`. Once Spring Security is in play it automatically publishes authentication events by default. This can be very useful for reporting, and also to implement a lock-out policy based on authentication failures.
Read more at **Spring Boot Actuator(Official Doc)**.

15. How to Use Jetty Instead of Tomcat in Spring-Boot-Starter-Web?

Remove the existing dependency on **spring-boot-starter-web** and add these in.

```
<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>
```

16. How to generate a WAR file with Spring Boot?

I suggest below three steps to generate and deployment Spring Boot WAR file.

1.Change the packaging type.

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

2.Add **spring-boot-starter-tomcat** as the **provided** scope

```
  <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-tomcat</artifactId>
      <scope>provided</scope>
  </dependency>
```

3.Spring Boot Application or *Main* class extends *SpringBootServletInitializer*

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
```

```
import org.springframework.boot.builder.SpringApplicationBuilder;
import org.springframework.boot.web.servlet.support.SpringBootServletInitializer;

@SpringBootApplication
public class Springboot2WebappJspApplication extends SpringBootServletInitializer{

    @Override
    protected SpringApplicationBuilder configure(SpringApplicationBuilder
application) {
        return application.sources(Springboot2WebappJspApplication.class);
    }

    public static void main(String[] args) {
        SpringApplication.run(Springboot2WebappJspApplication.class, args);
    }
}
```

Learn with a complete example at **Spring Boot 2 Deploy WAR file to External Tomcat**.

17. What is Spring Boot CLI(Command Line Interface) and it's Features?

The **Spring Boot CLI** is a *Command Line Interface* for Spring Boot. It can be used for a quick start with Spring. It can run Groovy scripts which means that a developer need not write boilerplate code; all that is needed is focus on business logic. **Spring Boot CLI** is the fastest way to create a Spring-based application.

Features

Let's look at the different features of Spring Boot CL –

•It provides an interface to run and test Spring Boot Application from a command prompt.

•It internally use Spring Boot Starter and Spring Boot AutoConfigurate components in order to resolve all dependencies and executes the application.

•It contains the Groovy compiler and Grape Dependency Manager.

•It supports Groovy Scripts without external Groovy installation.

•It adds Spring Boot defaults and resolve all dependencies automatically.

18. Where do you define properties in Spring Boot application?

You can define both application and Spring boot related properties into a file called **application.properties**. You can create this file manually or you can use Spring Initializer to create this file, albeit empty.

For example, by default, the embedded tomcat server start on port 8080 and by default, the context path is "/". Now let's change the default port and context path by defining properties in an **application.properties** file -

/src/main/resources/application.properties

```
server.port=8080
server.servlet.context-path=/springboot2webapp
```

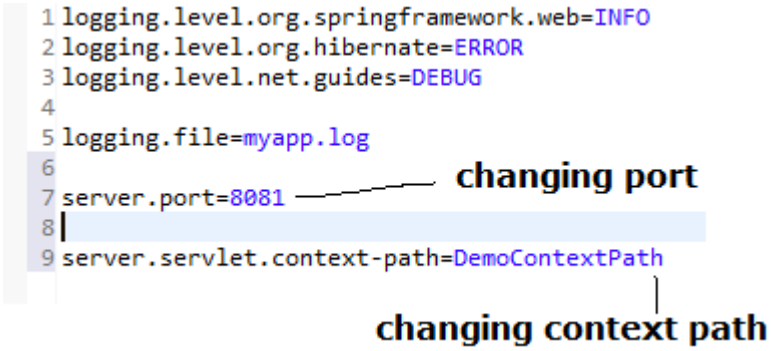
Read more at **Spring Boot How to Change Port and Context Path**.

19. How to Change Default Embedded Tomcat Server Port and Context Path in Spring Boot Application?

By default, the embedded tomcat server start on port 8080 and by default, the context path is "/". Now let's change the default port and context path by defining properties in an **application.properties** file -

/src/main/resources/application.properties

```
server.port=8080
server.servlet.context-path=/springboot2webapp
```



Read more at **Spring Boot How to Change Port and Context Path**.

20. What Embedded Containers Does Spring Boot Support?

Spring Boot support three embedded containers: Tomcat, Jetty, and Undertow. By default, it uses Tomcat as embedded containers but you can change it to Jetty or Undertow.

Spring Boot supports the following embedded servlet containers:

Name	Servlet Version
Tomcat 8.5	3.1
Jetty 9.4	3.1
Undertow 1.4	3.1

21. Can You Name Some Common Spring Boot Starter Poms?

I would like to list few commonly used Spring Boot Starter dependency in Spring boot applications.

- `spring-boot-starter-web` - Starter for building web, including RESTful, applications using Spring MVC. Uses Tomcat as the default embedded container

- `spring-boot-starter-data-jpa` - Starter for using Spring Data JPA with Hibernate

- `spring-boot-starter-test` - Starter for testing Spring Boot applications with libraries including JUnit, Hamcrest and Mockito

- `spring-boot-starter-thymeleaf` - Starter for building MVC web applications using Thymeleaf views

- `spring-boot-starter-data-mongodb` - Starter for using MongoDB document-oriented database and Spring Data MongoDB.

- `spring-boot-starter-security` - Starter for using Spring Security.

- `spring-boot-starter-actuator` - Starter for using Spring Boot's Actuator which provides production ready features to help you monitor and manage your application.

Read all Spring Boot starter dependencies at **Important Spring Boot Starters with Examples**.

22. How to use logging with Spring Boot?

We can use logging with Spring Boot by specifying log levels on **application.properties** file. Spring Boot loads this file when it exists in the classpath and it can be used to configure both Spring Boot and application code.

Spring Boot, by default, includes **spring-boot-starter-logging** as a transitive dependency for the **spring-boot-starter** module. By default, Spring Boot includes `SLF4J` along with `Logback` implementations.

If `Logback` is available, Spring Boot will choose it as the logging handler. You can easily configure logging levels within the application.properties file without having to create logging provider-specific configuration files such as `logback.xml` or `log4j.properties`.

```
logging.level.org.springframework.web=INFO
logging.level.org.hibernate=ERROR
```

```
logging.level.net.guides=DEBUG
```

Read more at **Spring Boot 2 Logging SLF4j Logback and LOG4j2 Example**.

23. What is the Spring Boot Starter Parent and How to Use it?

All Spring Boot projects typically use `spring-boot-starter-parent` as the parent in `pom.xml`.

```
    <parent>

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

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

        <version>2.0.4.RELEASE</version>

    </parent>
```

`spring-boot-starter-parent` allows us to manage the following things for multiple child projects and modules:

- Configuration - Java Version and Other Properties
- Dependency Management - Version of dependencies
- Default Plugin Configuration

We should need to specify only the Spring Boot version number on this dependency. If you import additional starters, you can safely omit the version number.

```
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.0.4.RELEASE</version>
    <relativePath /> <!-- lookup parent from repository -->
</parent>

<properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
    <java.version>1.8</java.version>
</properties>
                    override default Java version from parent

<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>   specify only the Spring Boot version number on this dependency
                    and  omit the version number for additional starters
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
                        omit version number
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-devtools</artifactId>
        <scope>runtime</scope>
    </dependency>
</dependencies>
```

Read more about spring-boot-starter-parent at **Overview of Spring Boot Starter Parent**.

24. How to Implement Security for Spring Boot Application ?

Spring boot provided auto-configuration of spring security for a quick start. Adding the Spring Security Starter (spring-boot-starter-security) to a Spring Boot application will:

- Enable HTTP basic security
- Register the `AuthenticationManager` bean with an in-memory store and a single user
- Ignore paths for commonly used static resource locations (such as /css/**, /js/**, /images/**, etc.)
- Enable common low-level features such as `XSS`, `CSRF`, caching, etc.

Add below dependencies to *`pom.xml`* file

```
<dependency>

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

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

</dependency>
```

Now if you run the application and access **http://localhost:8080**, you will be prompted to enter the user credentials. The default user is `user` and the password is auto-generated. You can find it in the console log.

Using default security password: `78fa095d-3f4c-48b1-ad50-e24c31d5cf35`

You can change the default user credentials in application.properties as follows:

```
security.user.name=admin
security.user.password=secret
security.user.role=USER,ADMIN
```

25. How to Configure Datasource Using Spring Boot?

Follow below steps to configure Datasource in Spring Boot applications:

1. Use either spring-boot-starter-jdbc or spring-boot-starterdata-jpa and include a JDBC driver on classpath
2. Declare properties in an application.properties

```
################### DataSource Configuration ##########################
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/test
spring.datasource.username=root
spring.datasource.password=root

spring.datasource.initialization-mode=always

################### Hibernate Configuration ##########################
```

```
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
```

– Spring Boot will create a DataSource with properties set

– Will even use a connection pool if the library is found on the classpath!