**What is Spring Boot and its need?**

Spring Boot is a Spring module that aims to simplify the use of the Spring framework for Java development. It is used to create stand-alone Spring-based applications that you can just run. So, it basically removes a lot of configurations and dependencies. Aiming at the Rapid Application Development, Spring Boot framework comes with the auto-dependency resolution, embedded HTTP servers, auto-configuration, management endpoints, and [Spring Boot CLI](https://www.edureka.co/blog/spring-boot-setup-helloworld-microservices-example/). Spring Boot not only improves productivity but also provides a lot of conveniences to write your own business logic. Spring Boot represents a fusion of the lightweight Spring application framework, configuration annotations, and embedded HTTP server. Made available with an auto-configuration feature, and support for Spring Initializer, Groovy, and Java, Spring Boot reduces Integration Test, Development, and Unit Test time. It aids the development of fast, responsive, and secure web applications, providing users with a complete configuration and programming model for Java enterprise applications. Spring Boot, utilizing the core features of the Spring application framework, offers a faster development technique for RESTful or REST architecture-based web services.

### How does Spring Boot works?

Spring Boot automatically configures your application based on the dependencies you have added to the project by using annotation. The entry point of the spring boot application is the class that contains @SpringBootApplication annotation and the main method. Spring Boot automatically scans all the components included in the project by using @ComponentScan annotation. Integrates with Spring Ecosystem like Spring [JDBC](https://www.edureka.co/blog/connect-mysql-database-in-java), Spring ORM, Spring Data, Spring Security easily by avoiding boilerplate code.

### ****Mention a few features of Spring Boot.****

Few important features of Spring Boot are as follows:

* **Spring CLI –** Spring Boot CLI allows you to Groovy for writing Spring boot application and avoids boilerplate code.
* **Starter Dependency** – With the help of this feature, Spring Boot aggregates common dependencies together and eventually improves productivity
* **Auto-Configuration** – The auto-configuration feature of Spring Boot helps in loading the default configurations according to the project you are working on. In this way, you can avoid any unnecessary WAR files.
* **Spring Initializer** – This is basically a web application, which can create an internal project structure for you. So, you do not have to manually set up the structure of the project, instead, you can use this feature.
* **Spring Actuator** –  This feature provides help while running Spring Boot applications.
* **Logging and Security** – The logging and security feature of Spring Boot, ensures that all the applications made using Spring Boot are properly secured without any hassle.

**What makes Spring Boot superior to JAX-RS?**

By leveraging Spring Boot features, users can experience significant advantages over JAX-RS, including:

* Fast deployment
* High scalability
* Container compatibility
* Minimal configuration
* Lower production time
* Increased productivity
* Reduced development time
* Easy monitoring and management of applications

**What Spring Boot features help develop Microservices Applications?**

Primarily used for developing microservices-based applications, Spring Boot offers the following key features for configuring, developing, and deploying microservices architecture.

* Integrates a tool called the Actuator, which enables users to manage and monitor applications
* Provides support for embedded servers, such as Jetty and Tomcat
* Users can simply run war files, without deploying it
* Includes an Auto-Configuration functionality, allowing users to configure Spring applications automatically
* Supports HTTP client Feign

**Why Spring Boot is preferred over any other framework?**

The Spring cloud that comes with Spring Boot, includes vast libraries, which is one of the major reasons why most developers prefer the Java-based Spring Boot. In addition, Spring Boot offers superior compatibility with Spring frameworks, and it also provides excellent support for docker containerization, heightening performance, and useability. Some of the most compelling reasons for using Spring Boot include:

* Provides the best means to configure Java beans
* Offers robust batch processing
* Helps users effectively manage Representational State Transfer (REST) endpoints
* Integrates an auto-configuration tool, eliminating the need for manual configuration
* Enables  annotation-based configurations
* Ease of dependency management
* Includes embedded servlet containers

### ****Mention the possible sources of external configuration.****

Spring Boot provides support for external configuration, allowing us to run the same application in various environments. This is done with the support it provides for external configuration. It uses environment variables, properties files, command-line arguments, YAML files, and system properties to mention the required configuration properties. Also, the **@value** annotation is used to gain access to the properties. **We can use properties files, YAML files, environment variables, system properties, and command-line option arguments to specify configuration properties.** We can then gain access to those properties using the @Value annotation, a bound object via the [**@ConfigurationProperties** annotation](https://www.baeldung.com/configuration-properties-in-spring-boot), or the Environment abstraction.

So, the most possible sources of external configuration are as follows:

* **Application Properties –** By default, Spring Boot searches for the application properties file or its YAML file in the current directory, classpath root or config directory to load the properties.
* **Command-line properties –** Spring Boot provides command-line arguments and converts these arguments to properties. Then it adds them to the set of environment properties.
* **Profile-specific properties –**  These properties are loaded from the application-{profile}.properties file or its YAML file. This file resides in the same location as that of the non-specific property files and the{profile} placeholder refers to an active profile.

### Can you explain what happens in the background when a Spring Boot Application is “Run as Java Application”?

When a Spring Boot application is executed as “Run as Java application”, then it automatically launches up the tomcat server as soon as it sees, that you are developing a web application.

**How to resolve whitelabel error page in spring boot application?**

This is quite common error in spring boot application which says 404(page not found).

We can mostly resolve this in 3 ways:

* **Custom Error Controller**– where you will be implementing ErrorController  interface which is provided by SpringFramework and then overriding its getErrorPath() so that you can return a custom path whenever such type of error is occurred.
* **By Displaying Custom error page**– All you have to do is create an error.html page and place it into the src/main/resources/templates path. The BasicErrorController of of springboot will automatically pick this file by default.
* **By disabling the whitelabel error page**– this is the easiest way where all you need to do is server.error.whitelabel.enabled property to false in the application.properties file to disable the whitelabel error page.

### ****What is response entity in spring boot?****

Response Entity is basically an HTTP response which includes headers,status code and body of your response.

### ****How to handle exceptions in spring boot?****

To handle exceptions in spring boot, you can use **@ControllerAdvice** annotation to handle your exceptions globally.

In order to handle specific exception and send customized response, you need to use **@ExceptionHandler** annotation.

### ****What is JPA in spring boot?****

JPA is basically **Java Persistence API**. It’s a specification that lets you do ORM when you are connecting to a relational database which is Object-Relational Mapping. So, when you need to connect from your java application to relational database, you need to be able to use something like JDBC and run SQL queries and then you get the results and convert them into Object instances. ORM lets you map your entity classes in your SQL tables so that when you connect to the database, you don’t need to do query yourself, it’s the framework that handles it for you. And JPA is a way to use ORM, it’s an API which lets you configure your entity classes and give it to a framework so that the framework does the rest.

### ****What are the Spring Boot starters and what are available the starters?****

Spring Boot starters are a set of convenient dependency management providers that can be used in the application to enable dependencies. These starters, make development easy and rapid. All the available starters come under the **org.springframework.boot** group. Few of the popular starters are as follows:

* spring-boot-starter: – This is the core starter and includes logging, auto-configuration support, and YAML.
* spring-boot-starter-jdbc – This starter is used for HikariCP connection pool with JDBC
* spring-boot-starter-web – Is the starter for building web applications, including RESTful, applications using Spring MVC
* spring-boot-starter-data-jpa – Is the starter to use Spring Data JPA with Hibernate
* spring-boot-starter-security – Is the starter used for Spring Security
* spring-boot-starter-aop: This starter is used for aspect-oriented programming with AspectJ and  Spring AOP
* spring-boot-starter-test: Is the starter for testing Spring Boot applications

### ****Explain Spring Actuator and its advantages.****

Actuator is one of the best parts of spring boot which consists of production-ready features to help you monitor and manage your application. With the help of an actuator, you can monitor what is happening inside the running application. Actuator dependency figures out the metrics and makes them available as a new endpoint in your application and retrieves all required information from the web. You can identify beans, the health status of your application, CPU usage, and many more with the actuator. There is no need to worry about security; if Spring Security is present, then these endpoints are secured by default using Spring Security’s content-negotiation strategy. Or else, we can configure custom security by the help of **RequestMatcher.** Essentially, Actuator brings Spring Boot applications to life by enabling production-ready features. These features allow us to monitor and manage applications when they're running in production. All we need to do is to include the spring-boot-starter-actuator starter in the pom.xml file:

<dependency>

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

<artifactId>**spring-boot-starter-actuator**</artifactId>

</dependency>

### ****How can we create a custom endpoint in Spring Boot Actuator?****

By using @Endpoint annotation, you can create a custom endpoint. **ReadOnly**

Spring Boot Actuator can expose operational information using either HTTP or JMX endpoints. Most applications go for HTTP, though, where the identity of an endpoint and the /actuator prefix form a URL path.

Here are some of the most common built-in endpoints Actuator provides:

* env: Exposes environment properties
* health: Shows application health information
* httptrace: Displays HTTP trace information
* info: Displays arbitrary application information
* metrics: Shows metrics information
* loggers: Shows and modifies the configuration of loggers in the application
* mappings: Displays a list of all @RequestMapping paths

**How to get the list of all the beans in your Spring boot application?**

Spring Boot actuator “/Beans” is used to get the list of all the spring beans in your application.

**How to check the environment properties in your Spring boot application?**

Spring Boot actuator “/env” returns the list of all the environment properties of running the spring boot application.

### How can we create a custom endpoint in Spring Boot Actuator?

To create a custom endpoint in Spring Boot 2.x, you can use the @Endpoint annotation. Spring Boot also exposes endpoints using @WebEndpointor, @WebEndpointExtension over HTTP with the help of [Spring MVC](https://www.edureka.co/blog/spring-mvc-tutorial/), [Jersey](https://www.edureka.co/blog/java-web-services-tutorial/), etc.

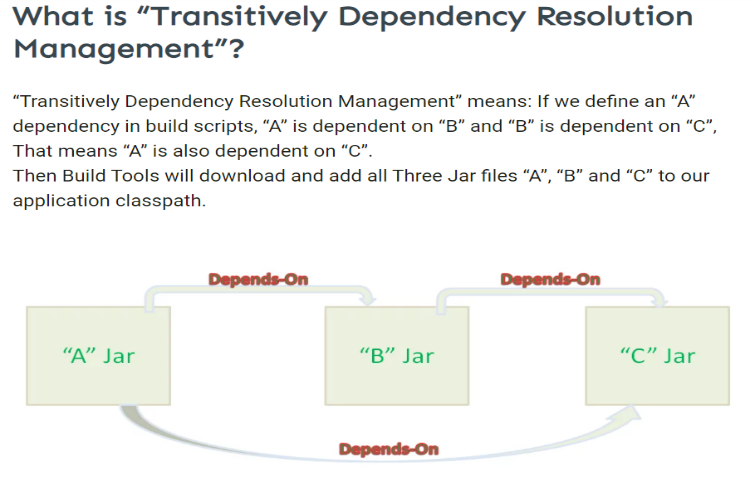
**What Are the Basic Annotations that Spring Boot Offers?**

The primary annotations that Spring Boot offers reside in its org.springframework.boot.autoconfigure and its sub-packages. Here are a couple of basic ones:

* @EnableAutoConfiguration – to make Spring Boot look for auto-configuration beans on its classpath and automatically apply them.
* @SpringBootApplication – used to denote the main class of a Boot Application. This annotation combines @Configuration, @EnableAutoConfiguration, and @ComponentScan annotations with their default attributes.

### ****What is Spring Boot dependency management?****

Spring Boot dependency management is basically used to manage dependencies and configuration automatically without you specifying the version for any of those dependencies.



**What is Spring Boot Initilizr?**

Spring Boot Initilizr is a Spring Boot tool to bootstrap Spring Boot or Spring Applications very easily. It comes in forms:

1. Spring Boot Initilizr With Web Interface
2. Spring Boot Initilizr With IDEs/IDE Plugins
3. Spring Boot Initilizr With Spring Boot CLI
4. Spring Boot Initilizr With ThirdParty Tools

**Why need?**

Spring Boot Initilizr simplifies Spring Applications Development by providing initial project structure and build scripts.

* It reduces Development time
* It increases Productivity

### ****Mention the minimum requirements for a Spring boot System.****

Spring Boot 2.1.7.RELEASE requires

* Java 8 +
* Spring Framework 5.1.9 +

**Explicit build support**

* Maven 3.3+
* Gradle 4.4+

**Servlet Container Support**

* Tomcat 9.0 – Servlet Version 4.0
* Jetty 9.4 –  Servlet Version 3.1
* Undertow 2.0 – Servlet Version 4.0

### ****Explain what is thymeleaf and how to use thymeleaf?****

Thymeleaf is a server-side java template engine which helps processing and creating HTML,XML, JavaScript , CSS, and text. Whenever the dependency in pom.xml(in case of  maven project) is added, springboot automatically configures Thymeleaf to serve dynamic web content. We can place the thyme leaf templates which are just the HTML files in **src/main/resources/templates/** folder so that spring boot can pick those files and renders whenever required. Thymeleaf will parse the index.html and will replace the dynamic values with its actual value that is been passed from the controller class. That’s it, once you run your Spring Boot application and your message will be rendered in web browsers. Thymeleaf is a server-side Java template engine used for web applications. It aims to bring natural template for your web application and can integrate well with Spring Framework and HTML5 Java web applications. To use Thymeleaf, you need to add the following code in the pom.xml file:

<dependency>

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

<artifactId>**spring-boot-starter-thymeleaf**</artifactId>

</dependency>

### ****Can we change the port of the embedded Tomcat server in Spring boot?****

We can [change the default port of a server embedded in Spring Boot](https://www.baeldung.com/spring-boot-change-port) using one of these ways:

* **using a properties file** – we can define this in an application.properties (or application.yml) file using the property server.port
* **programmatically** – in our main @SpringBootApplication class, we can set the server.port on the SpringApplication instance
* **using the command line** – when running the application as a jar file, we can set the server.port as a java command argument: java -jar -Dserver.port=8081 myspringproject.jar

### ****How to Write Integration Tests?****

When running integration tests for a Spring application, we must have an ApplicationContext. To make our life easier, Spring Boot provides a special annotation for testing – @SpringBootTest. This annotation creates an ApplicationContext from configuration classes indicated by its classes attribute**.** I**n case the**classes**attribute isn't set, Spring Boot searches for the primary configuration class.** The search starts from the package containing the test up until it finds a class annotated with @SpringBootApplication or @SpringBootConfiguration.

### ****What is the need for Spring Boot DevTools?****

Spring Boot Developer Tools, or DevTools, is a set of tools making the development process easier. To include these development-time features, we just need to add a dependency to the pom.xml file:

<dependency>

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

<artifactId>**spring-boot-devtools**</artifactId>

</dependency>

The spring-boot-devtools module is automatically disabled if the application runs in production. The repackaging of archives also excludes this module by default. Hence, it won't bring any overhead to our final product.

By default, DevTools applies properties suitable to a development environment. These properties disable template caching, enable debug logging for the web group, and so on. As a result, we have this sensible development-time configuration without setting any properties.

**Applications using DevTools restart whenever a file on the classpath changes.** This is a very helpful feature in development, as it gives quick feedback for modifications.By default, static resources, including view templates, don't set off a restart. Instead, a resource change triggers a browser refresh. Notice this can only happen if the LiveReload extension is installed in the browser to interact with the embedded LiveReload server that DevTools contains. This is one of the amazing features provided by Spring Boot, where it restarts the spring boot application whenever any changes are being made in the code. Here, you don’t need to right-click on the project and run your application again and again. Spring Boot dev tools does this for you with every code change.  
Themain focus of this module is to improve the development time while working on Spring Boot applications.

### ****Mention the steps to create a Spring Boot project using Spring  Initializer.****

Spring Initializr is a web tool provided by Spring. With the help of this tool, you can create Spring Boot projects by just providing project details. The Spring Initializer is a web application that generates a Spring Boot project with everything you need to start it quickly.

* Choose the maven project and the required dependencies. Then, fill in the other required details like Group, Artifact, and then click on Generate Project.
* Once the project is downloaded, extract the project onto your system
* Next, you have to import this project using the import option on the Spring Tool Suite IDE
* While importing the project, remember that you have to choose the project type to be Maven and the source project should contain the pom.xml file.

### ****How to enable HTTP/2 support in Spring Boot?****

You can enable the HTTP/2 support in Spring Boot by: **server.http2.enabled=true**

## ****What is a shutdown in the actuator?****

[Shutdown](https://docs.spring.io/spring-boot/docs/current/reference/html/production-ready-endpoints.html) is an endpoint that allows the application to be gracefully shutdown. This feature is not enabled by default. You can enable this by using **management.endpoint.shutdown.enabled=true** in your application.properties file.

## ****Can we disable the default web server in the Spring Boot application?****

The major strong point in Spring is to provide flexibility to build your application loosely coupled. Spring provides features to disable the web server in a quick configuration. Yes, we can use the application.properties to configure the web application type, i.e.  **spring.main.web-application-type=none.**

### What annotations are used to create an Interceptor?

A prominent functionality of Spring Boot, Interceptor uses the annotated class @Component, and it implements the interface HandlerInterceptor. The interface contains 3 main methods, which are:

* **The preHandle() Method** − preHandle() is used for intercepting the request prior to the implementation of the handler. If preHandle() returns a “true” boolean value, developers can continue with handler execution. If preHandle() returns a “false” boolean value, developers should stop the handler execution. preHandle() implementation looks like:

 @Override

    public boolean preHandle(HttpServletRequest httpServletRequest, HttpServletResponse httpServletResponse, Object o) throws Exception {

        logger.info(" Pre handle ");

        if(httpServletRequest.getMethod().equals("GET"))

            return true;

        else

return false;

    }

* **The postHandle() Method** − postHandle() is used for intercepting a request following the implementation of the handler. It allows the manipulation of the ModelAndView Object before users render it. postHandle() implementation looks like:

@Override

    public void postHandle(HttpServletRequest httpServletRequest, HttpServletResponse httpServletResponse, Object o, ModelAndView modelAndView) throws Exception {

        logger.info(" Post handle ");

        if(modelAndView.getModelMap().containsKey("status")){

            String status = (String) modelAndView.getModelMap().get("status");

            if(status.equals("SUCCESS!")){

                status = "Authentication " + status;

                modelAndView.getModelMap().put("status",status);

            }

        }

    }

* **The afterCompletion() Method** − A HandlerInterceptor callback approach, the afterCompletion() method is used when the entire request gets completed. afterCompletion() looks like:

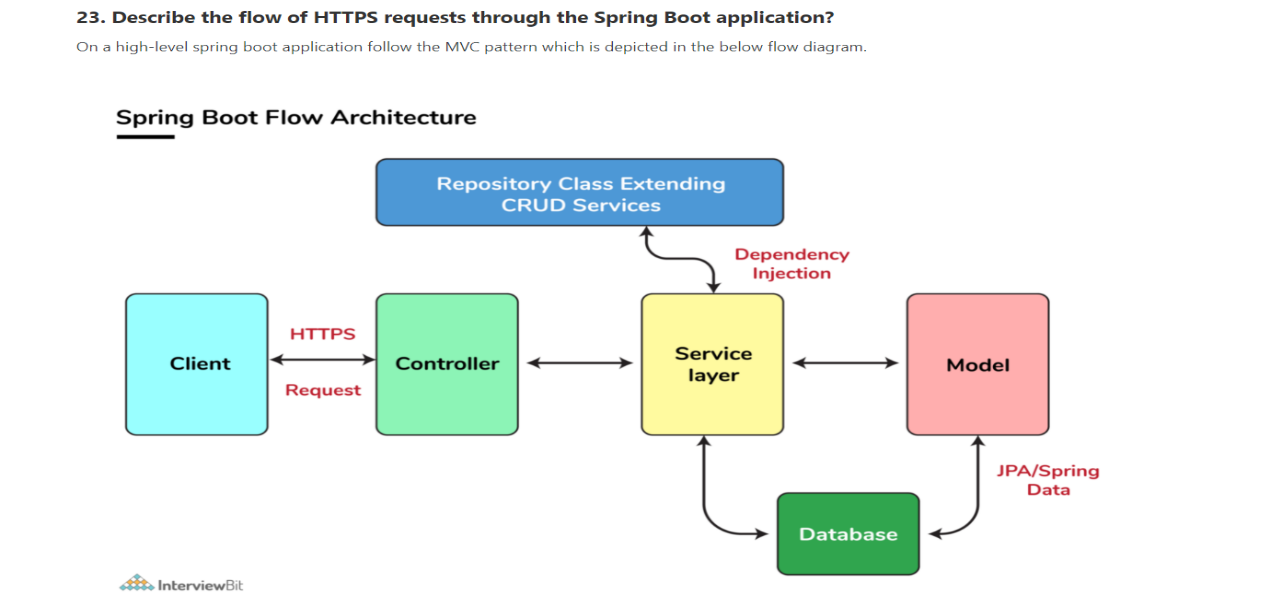
@Override

    public void afterCompletion(HttpServletRequest httpServletRequest, HttpServletResponse httpServletResponse, Object o, Exception e) throws Exception {

        logger.info(" After Completion ");

    }

}



**How to enable debugging log in the spring boot application?**

Debugging logs can be enabled in three ways -

* We can start the application with --debug switch.
* We can set the logging.level.root=debug property in application.property file.
* We can set the logging level of the root logger to debug in the supplied logging configuration file.

**Where do we define properties in the 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 use Spring Initializer to create this file. You don’t need to do any special configuration to instruct Spring Boot to load this file, If it exists in classpath then spring boot automatically loads it and configure itself and the application code accordingly.

### Which Embedded Servers does Spring Boot Support, and How to Change the Default?

As of date, **Spring MVC supports Tomcat, Jetty, and Undertow**. Tomcat is the default application server supported by Spring Boot's web starter. **Spring WebFlux supports Reactor Netty, Tomcat, Jetty, and Undertow** with Reactor Netty as default. In Spring MVC, to change the default, let's say to Jetty, we need to exclude Tomcat and include Jetty in the dependencies:

<**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**>

### How are the @RestController and @Controller Annotation different?

The traditional Spring @Controller annotation specifies that an annotated class represents a controller. It’s basically a @Component specialization, and it is autodetected via the classpath scanning. The @Controller annotation is used along with the annotated handler methodologies based on @RequestMapping annotations.

Developers use the @RestController annotation to develop RESTful web services, utilizing the Spring Model–View–Controller (MVC).  The Spring @RestController maps the request data to specified request handler methods. Once the handler method generates the response body, the @RestController modifies it to XML or JSON response.

@Controller Map of the model object to view or template and make it human readable but @RestController simply returns the object and object data is directly written in HTTP response as JSON or XML.

### ****What are the @RequestMapping  and @RestController annotation in Spring Boot used for?****

|  |  |
| --- | --- |
| **@RequestMapping** | **@RestController** |
| This annotation is used to provide the routing information and tells to Spring that any HTTP request must be mapped to the respective method. | This annotation is used to add the @ResponseBody and @Controller annotation to the class |
| To use this annotation, you have to import org.springframework.web.  bind.annotation.RequestMapping; | To use this annotation, you have to import org.springframework.web.  bind.annotation.RestController; |

### ****How to do pagination in spring boot?****

The process of dividing your data into small and suitable chunks is Pagination. One can achieve pagination by using PagingAndSortingRepository which is an extension of crudRepository.

**What are spring boot annotations**?

* Spring Framework Stereotype Annotations
* Core Spring Framework Annotations

### ****What is Spring Boot CLI and how to execute the Spring Boot project using boot CLI?****

Spring Boot CLI is nothing but a command-line tool which is provided by Spring so that you can develop your applications quicker and faster. Spring Boot CLI is a command-line interface that allows you to create a spring-based java application using Groovy. To execute your spring boot project using CLI, you need first to download CLI from Spring’s official website and extract those files. You may see a bin folder present in the Spring setup which is used to execute your spring boot application. As Spring boot CLI allows you to execute groovy files, you can create one and open it in the terminal. And then execute  **./spring run filename.groovy;**

With Spring Boot CLI:

* No Semicolons
* No Public and private access modifiers
* No Imports(Most)
* No “return” statement
* No setters and getters
* No Application class with main() method(It takes care by SpringApplication class).
* No Gradle/Maven builds.
* No separate HTTP Servers.

Spring Boot CLI is a tool supported by the official [Spring Framework](https://www.edureka.co/blog/what-is-spring-framework/). The steps to execute a Spring Boot project are as follows:

* Download the CLI tool from the official site and extract the zip file. The bin folder present in the Spring setup is used to execute the Spring Boot application.
* Since Spring Boot CLI executes groovy files, you need to create a groovy file for Spring Boot application. So, to do that, open terminal and change the current directory to the bin folder. Now, open a groovy file (for example Sample.groovy)
* In this file create a controller as follows:

@RestController public class Sample {

@RequestMapping("/example")

String index(){

<h1>"Welcome To Edureka"</h1>;

} }

* Then execute the groovy file by mentioning: ./spring run Sample.groovy;

Once, the project is executed go to the URL(localhost:8080:/example) and you will see the output as **Welcome To Edureka**

### What are the most common Spring Boot CLI commands?

-run, -test, -grap, -jar, -war, -install, -uninstall, --init, -shell, -help.

### ****How to run spring-boot jar from the command line?****

In order to run spring boot jar from the command line, you need to update you pom.xml(or build.gradle) of your project with the maven plugin.

### <build>

### <plugins>

### <plugin>

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

### <artifactId>spring-boot-maven-plugin</artifactId>

### </plugin>

### </plugins>

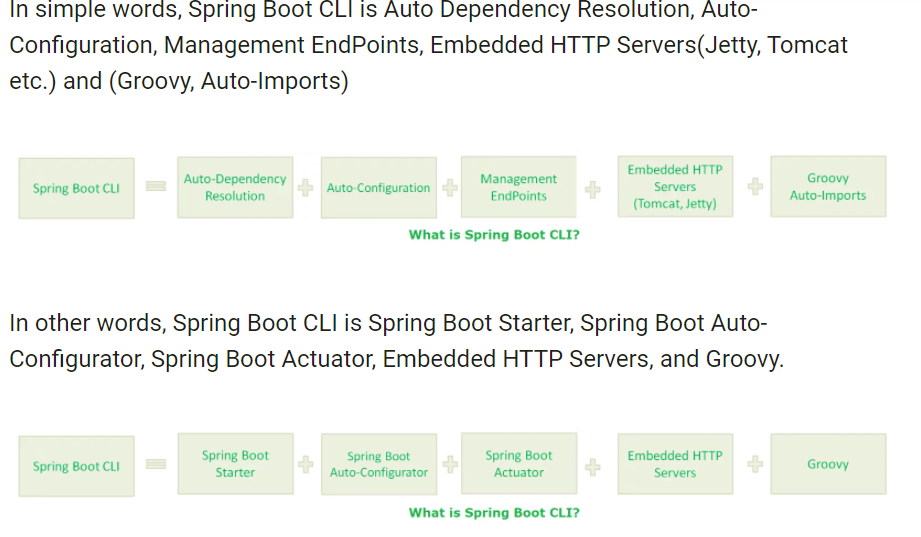
### </build>

Now, Build your application and package it into the single executable jar. Once the jar is built you can run it through the command prompt  using the below query:

java -jar target/myDemoService-0.0.1-SNAPSHOT.jar

And you have your application running.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### ****Mention the differences between JPA and****[Hibernate.](https://www.edureka.co/blog/what-is-hibernate-in-java/)

|  |  |
| --- | --- |
| **JPA** | **Hibernate** |
| JPA is a Data Access Abstraction used to reduce the amount of boilerplate code | Hibernate is an implementation of Java Persistence API and offers benefits of loose coupling |

A Java Persistence API (JPA) implementation, Hibernate facilitates Object-Relational Mapping (ORM), allowing users to store, retrieve, map, and update application data to and from Java objects and relational databases. Hibernate maps the data types in Java to SQL (Structured Query Language) data types, and the classes in java to the database tables, relieving developers from scripting data persistence SQL programs.

A Spring Data sub-project, Spring Data JPA, on the other hand, gives abstraction over the DAL (Data Access Layer) applying JPA and Object–Relational Mapping implementations, such as Hibernate. Spring Data JPA facilitates the smooth implementation of JPA repositories, and it intends to improve the overall implementation of DAL to a great extent.

### ****Explain Spring Data.****

Spring Data aims to make it easy for the developers to use relational and non-relational databases, cloud-based data services, and other data access technologies. So, basically, it makes it easy for data access and still retains the underlying data.

### What do you understand by auto-configuration in Spring Boot and how to disable the auto-configuration?

Auto-configuration is used to automatically configure the required configuration for the application. For example, if you have a data source bean present in the classpath of the application, then it automatically configures the [JDBC template](https://www.edureka.co/blog/connect-mysql-database-in-java). With the help of auto-configuration, you can create a Java application in an easy way, as it automatically configures the required beans, controllers, etc.

To disable the auto-configuration property, you have to exclude attribute of @EnableAutoConfiguration, in the scenario where you do not want it to be applied.

**@EnableAutoConfiguration(exclude={DataSourceAutoConfiguration.class})**

If the class is not on the classpath, then to exclude the auto-configuration, you have to mention the following code:

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

Apart from this, Spring Boot also provides the facility to exclude list of auto-configuration classes by using the spring.autoconfigure.exclude property. You can go forward, and add it either in the application.properties or add multiple classes with comma-separated.

### What are the differences between @SpringBootApplication and @EnableAutoConfiguration annotation?

|  |  |
| --- | --- |
| **@SpringBootApplication** | **@EnableAutoConfiguration** |
| Used in the main class or bootstrap class | Used to enable auto-configuration  and component scanning in your project |
| It is a combination of @Configuration, @ComponentScan and @EnableAutoConfiguration annotations. | It is a combination of @Configuration and @ComponentScan annotations |

## ****What does the @SpringBootApplication annotation do internally?****

The @SpringBootApplication annotation is equivalent to using @Configuration, @EnableAutoConfiguration, and @ComponentScan with their default attributes. Spring Boot enables the developer to use a single annotation instead of using multiple. But, as we know, Spring provided loosely coupled features that we can use for each annotation as per our project needs.

### How does a spring boot application get started?

Just like any other Java program, a Spring Boot application must have a main method. This method serves as an entry point, which invokes the SpringApplication#run method to bootstrap the application.

@SpringBootApplication

**public** **class** **MyApplication** {

**public** **static** **void** **main**(String[] args) {

SpringApplication.run(MyApplication.class);

// other statements

}

}

### What is the purpose of using @ComponentScan in the class files?

Spring Boot application scans all the beans and package declarations when the application initializes. You need to add the @ComponentScan annotation for your class file to scan your components added to your project.

### What are the steps to deploy Spring Boot web applications as JAR and WAR files?

To deploy a Spring Boot web application, you just have to add the following plugin in the pom.xml file:

<plugin>

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

    <artifactId>**spring-boot-maven-plugin**</artifactId>

</plugin>

By using the above plugin, you will get a JAR executing the package phase. This JAR will contain all the necessary libraries and dependencies required. It will also contain an embedded server. So, you can basically run the application like an ordinary JAR file.

**\*\*\*\* Note:** The packaging element in the pom.xml file must be set to **jar** to build a JAR file as below:

**<packaging>jar</packaging>**

Similarly, if you want to build a WAR file, then you will mention

**<packaging>war</packaging>**

* Generate a WAR from the project
* Then, deploy the WAR file onto your favorite server

\*\* Note: The steps to deploy the WAR file on the server is dependent on the server you choose.

### What is the best way to expose custom application configuration with Spring Boot?

One way to expose the custom application [configuration in Spring](https://www.edureka.co/blog/spring-tutorial/) Boot is by using the **@Value annotation**. But, the only problem with this annotation is that all the configuration values will be distributed throughout the application. Instead, you can use a centralized approach. By centralized approach, you can define a configuration component using the @ConfigurationProperties as follows:

@Component

@ConfigurationProperties("**example**")

public class SampleConfiguration {

private int number;

private boolean value;

private String message;

According to the above snippet, the values configured in application.properties will be as follows:

**example.number: 100**

**example.value: true**

**example.message: Dynamic Message**

### ****Can we create a non-web application in Spring Boot?****

Yes, we can create a non-web application by removing the web dependencies from the classpath along with changing the way Spring Boot creates the application context.

### What are the steps to connect an external database like MySQL or Oracle?

To connect an external database, you have to follow the below steps:

* Start by adding the dependency for MySQL Connector to pom.xml
* Then remove H2 Dependency from pom.xml
* Now, set up your [MySQL database](https://www.edureka.co/blog/mysql-tutorial/) and configure your connection to the MySQL database
* Restart your project

### Mention the advantages of the YAML file than Properties file and the different ways to load YAML file in Spring boot.

YAML gives you more clarity and is very friendly to humans. It also supports **maps, lists, and other scalar types.**

YAML comes with hierarchical nature which helps in avoiding repetition as well as indentations.

If we have different deployment profiles such as  development, testing, or production and we may have different configurations for each environment, so instead of creating new files for each environment we can place them in a single YAML file. But in the case of the properties file, you cannot do that. The advantages of the YAML file than a properties file is that the data is stored in a hierarchical format. So, it becomes very easy for the developers to debug if there is an issue. The SpringApplication class supports the YAML file as an alternative to properties whenever you use the SnakeYAML library on your classpath. The different ways to load a YAML file in Spring Boot is as follows:

* Use YamlMapFactoryBean to load YAML as a Map
* Use YamlPropertiesFactoryBean to load YAML as Properties



### How is Hibernate chosen as the default implementation for JPA without any configuration?

When we use the Spring Boot Auto Configuration, automatically the spring**-boot-starter-data-jpa**dependency gets added to the pom.xml file. Now, since this dependency has a transitive dependency on JPA and Hibernate, Spring Boot automatically auto-configures Hibernate as the default implementation for JPA, whenever it sees Hibernate in the classpath.

### What do you understand by Spring Data REST?

Spring Data REST is used to expose the RESTful resources around Spring Data repositories. Consider the following **example:**

@RepositoryRestResource(collectionResourceRel = "sample", path = "sample")

public interface SampleRepository

        extends CustomerRepository<sample, Long> {

**Now, to expose the REST services, you can use the POST method in the following way:**

{

"customername": "Rohit"

}

**Response Content**

{

"customername": "Rohit"

"\_links": {

"self": {

"href": "<a href="http://localhost:8080/sample/1">[http://localhost:8080/sample/1</a](http://localhost:8080/sample/1%3c/a)>" }

Observe that the response content contains the href of the newly created resource.

### What is the difference between RequestMapping and GetMapping?

The @GetMapping is a composed annotation that acts as a shortcut for @RequestMapping(method = RequestMethod.GET). Both these methods support the consumes. The consume options are :

consumes = “text/plain”  
consumes = {“text/plain”, “application/\*”}

### How does path=”sample”, collectionResourceRel=”sample” work with Spring Data Rest?

@RepositoryRestResource(collectionResourceRel = "sample", path = "sample")

public interface SampleRepository extends

### PagingAndSortingRepository<Sample, Long>

* path – This section is used to mention the segment under which the resource is to be exported.
* collectionResourceRel – This value is used to generate links to the collection resource.

### In which layer, should the boundary of a transaction start?

The boundary of the transaction should start from the Service Layer since the logic for the business transaction is present in this layer itself.

### Explain how to register a custom auto-configuration.

In order to register an auto-configuration class, you have to mention the fully-qualified name under the @EnableAutoConfiguration key META-INF/spring. factories file. Also, if we build the with maven, then this file should be placed in the resources/META-INT directory.

### ****How do you Configure Log4j for logging?****

Since Spring Boot supports Log4j2 for logging a configuration, you have to exclude Logback and include Log4j2 for logging. This can be only done if you are using the starters project.

### What is a Swagger in Spring Boot?

Swagger is used for clearly detailing and documenting RESTful APIs in a machine-readable and human-readable format, which is easily comprehensible for testers and developers, as well as individuals having little knowledge of source code. Enabling hassle-free application discovery, development, and integration, Swagger allows API consumers to interact with remote services with minimum implementation logic.

### ****Mention the differences between WAR and embedded containers****

|  |  |
| --- | --- |
| WAR | Embedded Containers |
| WAR benefits a considerable measure from Spring Boot | Only one component of Spring Boot and is utilized during improvements |

### What do you think is the need for Profiles?

### Profiles are used to provide a way to segregate the different parts of the application configuration and make it available for various environments. So, basically, any @Component or a @Configuration can be marked with a @Profile to limit as it is loaded. The application has different stages-such as the development stage, testing stage, production stage and may have different configurations based on the environments. With the help of spring boot, you can place profile-specific properties in different files such as application-{profile}.properties. In the above, you can replace the profile with whatever environment you need, for example, if it is a development profile, then application-development.properties file will have development specific configurations in it. So, in order to have profile-specific configurations/properties, you need to specify an active profile.

Consider you have multiple environments,

* Dev
* QA
* Stage
* Production

Now, let’s say, you want to have different application configuration in each of the environments, you can use profiles to have different application configurations for different environments. So, basically, Spring and Spring Boot provide features through which you can specify:

* The active profile for a specific environment
* The configuration of various environments for various profiles.

Once you are done with the profile-specific configuration, you have to set the active profile in an environment. To do that, either you can

* Use -Dspring.profiles.active=prod in  arguments
* Use spring.profiles.active=prod in application.properties file

### ****How to call servlet in spring boot?****

Servlet can be configured by using implementing or extending WebAppInitializer interface, AbstractDispatcherServletInitializer abstract class, and AbstractAnnotationConfigDispatcherServletInitializer abstract class

### ****What is the HTTP methods that can be implemented in spring boot rest service?****

Spring boot rest service helps in CRUD (create,retrieve,update,delete)operations. So based on that most commonly used HTTP methods are GET, POST, PUT, DELETE, and PATCH are the methods that can be implemented in spring boot rest services.

### ****How to debug spring boot application in eclipse?****

Debugging your spring boot application is as simple as same as debugging your Java application.  
All you have to do is place debug point within your application, and right-click-> click on debug as java application or spring boot application.

### ****How to insert data in mysql using spring boot?****

First configure mysql in your spring boot application.

Then you can map your entities with your db tables using JPA.

And with the help of save() method in JPA, you can directly insert your data into your database.

### ****How to create a login page in spring boot?****

You can create a simple and default login page in spring boot, you can make use of Spring security. Spring security secures all HTTP endpoints where the user has to login into the default HTTP form provided by spring.

We need to add **spring-boot-starter-security** dependency in your pom.xml or build.gradle and a default username and password can be generated with which you can log in.

### ****How to ignore null values in JSON response spring boot?****

In order to ignore null values in JSON response, you can use **@JsonIgnore** annotation.  
The field can be ignored while reading JSON into Java objects as well as while writing Java objects into JSON.

### ****What is the main class in spring boot?****

Usually in java applications, a class that has a main method in it is considered as a main class. Similarly, in spring boot applications main class is the class which has a public static void main() method and which starts up the SpringApplicationContext.

### What are the steps to add a custom JS code with Spring Boot?

All you have to do is just create a static folder under **src/main/resources** and place all your **static content** in this folder. And your custom JS code will be added with spring boot. The steps to add a [custom JS code](https://www.edureka.co/blog/javascript-tutorial/) with Spring Boot are as follows:

* Now, create a folder and name it **static** under the resources folder
* In this folder, you can put the static content in that folder

**Note:** Just in case, the browser throws an unauthorized error, you either disable the security or search for the password in the log file, and eventually pass it in the request header.

### How to instruct an auto-configuration to back off when a bean exists?

To instruct an auto-configuration class to back off when a bean is already existent, we can use the @ConditionalOnMissingBean annotation. The most noticeable attributes of this annotation are:

* value: The types of beans to be checked
* name: The names of beans to be checked

When placed on a method adorned with @Bean, the target type defaults to the method's return type:

@Configuration

**public** **class** **CustomConfiguration** {

@Bean

@ConditionalOnMissingBean

**public** CustomService **service**() { ... }

}

To instruct an auto-configuration class to back off when a bean exists, you have to use the @ConditionalOnMissingBean annotation. The attributes of this annotation are as follows:

* **value:** This attribute stores the type of beans to be checked
* **name:** This attribute stores the name of beans to be checked

### Why is Spring Data REST not recommended in real-world applications?

By using Spring Data Rest, you have access to all the RESTful resources that revolves around Spring Data repositories.Spring Data REST is not recommended in real-world applications as you are exposing your database entities directly as [REST Services](https://www.edureka.co/blog/what-is-rest-api/). While designing RESTful services, the two most important things that we consider is the domain model and the consumers. But, while using Spring Data REST, none of these parameters are considered. The entities are directly exposed. So, I would just say, you can use Spring Data REST, for the initial evolution of the project.

### What is the error you see if  H2 is not in the classpath?

If H2 is not present in the classpath, then you see the following error:

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

To resolve this error, add H2 to the pom.xml file, and restart your server.  
The following code snippet can be added to add the dependency:

<dependency>

    <groupId>**com.h2database**</groupId>

    <artifactId>**h2**</artifactId>

    <scope>runtime</scope>

</dependency>

### Mention the dependencies needed to start up a JPA Application and connect to in-memory database H2 with Spring Boot?

The dependencies are needed to start up a JPA Application and connect to in-memory database H2 with Spring Boot

* web starter
* h2
* data JPA starter
* To include the dependencies refer to the following code:

<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>

### What do you understand by Spring Boot supports relaxed binding?

Relaxed binding, is a way in which, the property name does not need to match the key of the environment property. In Spring Boot, relaxed binding is applicable to the type-safe binding of the configuration properties. For example, if a property in a bean class with the @ConfigurationPropertie annotation is used sampleProp, then it can be bounded to any of the following environment properties:

* sampleProp
* sample-Prop
* sample\_Prop
* SAMPLE\_PROP

### Where is the database connection information specified and how does it automatically connect to H2?

Well, the answer to this question is very simple. It is because of the Spring Boot auto-configuration that, configures the dependencies of the application. So, the database connection information, and automatically connecting the database to H2 is done by the auto-configuration property.

### What is the name of the default H2 database configured by Spring Boot?

The name of the default H2 database is **testdb.  Refer below:**

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

**Note:** Just incase if you are using H2 in-memory database, then exactly that is the name of Spring Boot which is used to setup your H2 database.

### ****Explain how to create a Spring Boot application using Maven.****

We can include Spring Boot in a Maven project just like we would any other library. However, the best way is to inherit from the spring-boot-starter-parent project and declare dependencies to [Spring Boot starters](https://www.baeldung.com/spring-boot-starters). Doing this lets our project reuse the default settings of Spring Boot.

Inheriting the spring-boot-starter-parent project is straightforward – we only need to specify a parent element in pom.xml:

<**parent**>

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

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

<**version**>2.4.0.RELEASE</**version**>

</**parent**>

We can find the latest version of spring-boot-starter-parent on [Maven Central](https://search.maven.org/search?q=g:org.springframework.boot%20AND%20a:spring-boot-starter-parent&core=gav).

**Using the starter parent project is convenient, but not always feasible.** For instance, if our company requires all projects to inherit from a standard POM, we can still benefit from Spring Boot's dependency management using a [custom parent](https://www.baeldung.com/spring-boot-dependency-management-custom-parent).

Well, there are various approaches to [create a Spring Boot application](https://www.edureka.co/blog/microservices-with-spring-boot) using maven, but if I have to name a few, then following are the ways to create a Spring Boot project/ application using [maven](https://www.edureka.co/blog/maven-in-java/):

* Spring Boot CLI
* Spring Starter Project Wizard
* Spring Initializr
* Spring Maven Project

**What is the difference between Spring and Spring Boot**

**Spring –**

1. Is a dependency injection framework.
2. It is basically used to manage the life cycle of java classes (beans). It consists of a lot of boilerplate configuration.
3. Uses XML based configuration.
4. It takes time to have a spring application up and running and it’s mainly because of boilerplate code.

**Spring boot- 2.5.0 (latest)**

1. It is a suite of pre- configured frameworks and technologies which helps to remove boilerplate configuration.
2. Uses annotations.
3. It is used to create a production-ready code.

|  |  |
| --- | --- |
| **Spring** | **Spring** Boot |
| A web application framework based on Java | A module of Spring |
| Provides tools and libraries to create customized web applications | Used to create a Spring application project which can just run/ execute |
| Spring is more complex than Spring Boot | Spring Boot is less complex than the Spring framework |
| Takes an unopinionated view | Takes an opinionated view of a platform |
|  |  |
|  |  |
|  |  |
|  |  |

### ****Mention the steps to connect Spring Boot application to a database using JDBC.****

Spring Boot starter projects provide the required libraries to connect the application with JDBC. So, for example, if you just have to create an application  and connect it with [MySQL](https://www.edureka.co/blog/mysql-tutorial/) database, you can follow the below steps:

**Step 1:** Create a database in MySQL

CREATE DATABASE example;

**Step 2:**Then you have to create a table inside this database.

CREATE TABLE customers(customerid INT PRIMARY KEY NOT NULL AUTO\_INCREMENT, customername VARCHAR(255));

**Step 3:** Now, create a Spring Boot project and provide the required details

**Step 4:** Add the JDBC, MySQL and web dependencies.

**Step 5:** Once the project is created, you have to configure the database into application properties

spring.datasource.url=jdbc:mysql://localhost:3306/example

spring.datasource.username=root

spring.datasource.password=edureka

spring.jpa.hibernate.ddl-auto=create-drop

**Step 6:** The main application.java class should have the following code:

package com.edureka;

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:** Next, you have to create a controller to handle the HTTP requests, by mentioning the following code:

package com.edureka;

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:** Finally, execute this project as a Java application.  
**Step 9:** Next, open the URL (localhost:8080/insert), and you will see the output as Data Entry Successful. You can also go forward and check if the data is entered into the table.