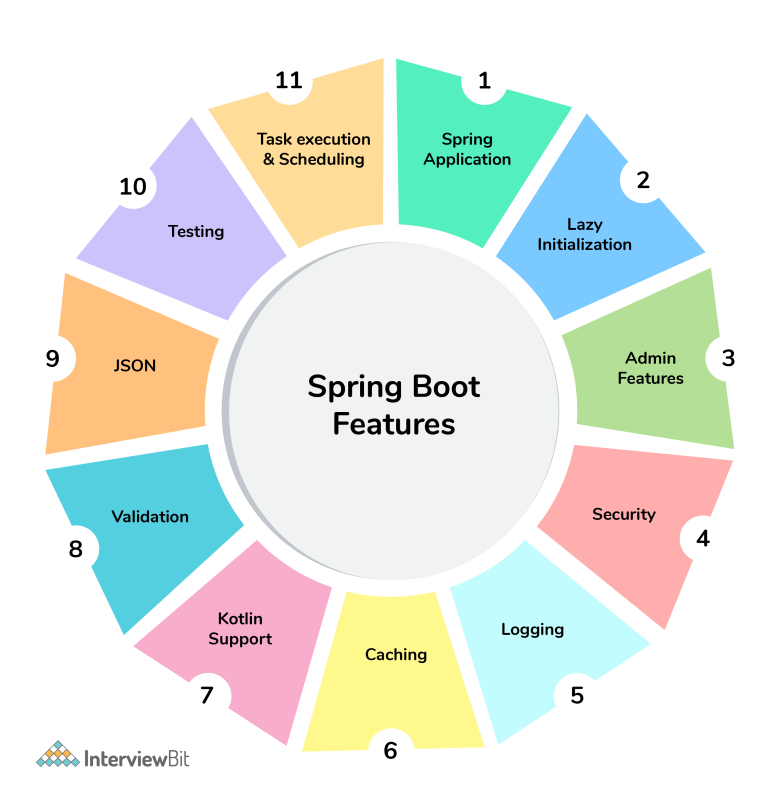
**What is Spring Boot?**

|  |
| --- |
| Spring Boot is a Spring module which provides RAD (Rapid Application Development) feature to Spring framework. |
| It is used to create stand alone spring based application that you can just run because it needs very little spring configuration. |
| Sprint boot is a Java-based spring framework used for Rapid Application Development (to build stand-alone microservices). It has extra support of auto-configuration and embedded application server like tomcat, jetty, etc. |
| Spring Boot is called a microservice framework that is built on top of the spring framework. This can help developers to focus more on convention rather than configuration. |
| The main aim of Spring boot is to give you a production-ready application. So, the moment you create a spring-boot project, it is runnable and can be executed/deployed on the server. |
| It comes with features like autoconfiguration, auto dependency resolution, embedded servers, security, health checks which enhances the productivity of a developer. |

**Features of Spring Boot that make it different?**

* Creates stand-alone spring application with minimal configuration needed.
* It has embedded tomcat, jetty which makes it just code and run the application.
* Provide production-ready features such as metrics, health checks, and externalized configuration.
* Absolutely no requirement for XML configuration.



**What are the advantages of using Spring Boot?**

The advantages of Spring Boot are listed below:

* Easy to understand and develop spring applications.
* Spring Boot is nothing but an existing framework with the addition of an embedded HTTP server and annotation configuration which makes it easier to understand and faster the process of development.
* Increases productivity and reduces development time.
* Minimum configuration.
* We don’t need to write any XML configuration, only a few annotations are required to do the configuration.

**What are the Spring Boot key components?**

Below are the four key components of spring-boot:

* Spring Boot auto-configuration.
* Spring Boot CLI.
* Spring Boot starter POMs.
* Spring Boot Actuators.

**Why Spring Boot over Spring?**

Below are some key points which spring boot offers but spring doesn’t:

* Starter POM.
* Version Management.
* Auto Configuration.
* Component Scanning.
* Embedded server.
* InMemory DB.
* Actuators
* Spring Boot simplifies the spring feature for the user:

**What are the advantages of Spring Boot?**

* Create stand-alone Spring applications that can be started using java -jar.
* Embed Tomcat, Jetty or Undertow directly. You don't need to deploy WAR files.
* It provides opinionated 'starter' POMs to simplify your Maven configuration.
* It automatically configure Spring whenever possible.

**How to create Spring Boot application using Maven?**

There are multiple approaches to create Spring Boot project. We can use any of the following approach to create application.

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

**How to create Spring Boot project using Spring Initializer?**

It is a web tool which is provided by Spring on official site. You can create Spring Boot project by providing project details.

**How to create Spring Boot project using boot CLI?**

It is a tool which you can download from the official site of Spring Framework. Here, we are explaining steps.

**How to create simple Spring Boot application?**

To create an application. We are using STS (Spring Tool Suite) IDE and it includes the various steps that are explaining in steps.

**What is Spring Boot dependency management?**

Spring Boot manages dependencies and configuration automatically. You don't need to specify version for any of that dependencies.

Spring Boot upgrades all dependencies automatically when you upgrade Spring Boot.

**What are the Spring Boot properties?**

Spring Boot provides various properties which can be specified inside our project's application. properties file. These properties have default values and you can set that inside the properties file. Properties are used to set values like: server-port number, database connection configuration etc.

**What are the Spring Boot Starters?**

Starters are a set of convenient dependency descriptors which we can include in our application.

Spring Boot provides built-in starters which makes development easier and rapid. For example, if we want to get started using Spring and JPA for database access, just include the spring-boot-starter-data-jpa dependency in your project.

**What is Spring Boot Actuator?**

Spring Boot provides actuator to monitor and manage our application. Actuator is a tool which has HTTP endpoints. when application is pushed to production, you can choose to manage and monitor your application using HTTP endpoints.

**What is thymeleaf?**

It is a server side Java template engine for web application. It's main goal is to bring elegant natural templates to your web application.

It can be integrate with Spring Framework and ideal for HTML5 Java web applications.

**How to use thymeleaf?**

In order to use Thymeleaf we must add it into our pom.xml file like:

<dependency>

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

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

</dependency>

**How to connect Spring Boot to the database using JPA?**

Spring Boot provides spring-boot-starter-data-jpa starter to connect Spring application with relational database efficiently. You can use it into project POM (Project Object Model) file.

**How to connect Spring Boot application to database using JDBC?**

Spring Boot provides starter and libraries for connecting to our application with JDBC. Here, we are creating an application which connects with Mysql database. It includes the following steps to create and setup JDBC with Spring Boot.

**What is @RestController annotation in Spring Boot?**

The @RestController is a stereotype annotation. It adds @Controller and @ResponseBody annotations to the class. We need to import org.springframework.web.bind.annotation package in our file, in order to implement it.

**What is @RequestMapping annotation in Spring Boot?**

The @RequestMapping annotation is used to provide routing information. It tells to the Spring that any HTTP request should map to the corresponding method. We need to import org.springframework.web.annotation package in our file.

**How to create Spring Boot application using Spring Starter Project Wizard?**

There is one more way to create Spring Boot project in STS (Spring Tool Suite). Creating project by using IDE is always a convenient way. Follow the following steps in order to create a Spring Boot Application by using this wizard.

**Spring Vs Spring Boot?**

Spring is a web application framework based on Java. It provides tools and libraries to create a complete cutomized web application.

Wheras Spring Boot is a spring module which is used to create spring application project that can just run.

**How to deploy spring boot application in tomcat?**

Whenever you will create your spring boot application and run it, Spring boot will automatically detect the embedded tomcat server and deploy your application on tomcat.

After successful execution of your application, you will be able to launch your rest endpoints and get a response.

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

**Spring –**

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

**Spring boot-**

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

**What is actuator in spring boot?**

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

**How to change port in spring boot?**

The default port number to start your SpringBoot application is 8080.

Just to change the port number, you need to add server.port=8084 (your port number) property in your application.properties file and start your application.

**How to create war file in spring boot?**

To create a war file in spring boot you need to define your packaging file as war in your pom.xml(if it is maven project).

Then just do maven clean and install so that your application will start building. Once the build is successful, just go into your Target folder and you can see .war file generated for your application.

**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.

How to save image in database 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

@RestController

@RequestMapping("/greatleasrning")

public class Controller {

@Autowired

private final GreatLearningRepository greatLearningRepository;

public Controller(GreatLearningRepository greatLearningRepository) {

}

In above case, your data which may be in JSON format can be inserted successfully into database.

@RequestMapping(method = RequestMethod.POST)

ResponseEntity<?> insert(@RequestBody Course course) {

greatLearningRepository.save(course);

return ResponseEntity.accepted().build(); } }

**What is auto configuration in spring boot?**

AutoConfiguration is a process by which Spring Boot automatically configures all the infrastructural beans. It declares the built-in beans/objects of the spring specific module such as JPA, spring security and so on based on the dependencies present in your applications class path.

**For example:** If we make use of Spring JDBC, the spring boot autoconfiguration feature automatically registers the DataSource and JDBCTemplete bean.

This entire process of automatically declaring the framework specific bean without the need of writing the xml code or java config code explicity is called Autoconfiguration which is done by springBoot with the help of an annotation called @EnableAutoconfiguration alternatively @SpringBootApplication.

**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.

**How to fetch data from database in spring boot?**

You can use the following steps to connect your application with MySQL database.

1. First create a database in MySQL with create DATABASE student;

2. Now, create a table inside this DB:

CREATE TABLE student(studentid INT PRIMARY KEY NOT NULL AUTO\_INCREMENT, studentname VARCHAR(255));

3. Create a SpringBoot application and add JDBC, MySQL and web dependencies.

4. In application.properties, you need to configure the database.

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

spring.datasource.username=system123

spring.datasource.password=system123

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

5. In your controller class, you need to handle the requests.

package com.student;

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("/save")

public String index(){

jdbc.execute("insert into student (name)values(GreatLearnings)");

return "Data Entry Successful"; } }

6. Run the application and check the entry in your Database.

**How to use logger in spring boot?**

There are many logging options available in SpringBoot. Some of them are mentioned below:

**Using log4j2:**

import org.apache.logging.log4j.Logger;

import org.apache.logging.log4j.LogManager;

// [...]

Logger logger = LogManager.getLogger(LoggingController.class);

**Using Lombok:**

All you need to do is add a dependency called org.projectlombok in your pom.xml as shown below:

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<version>1.18.4</version>

</dependency>

Now you can create a loggingController and add the @Slf4j annotation to it. Here we would not create any logger instances.

@RestController

@Slf4j

public class LoggingController {

@RequestMapping("/logging")

public String index() {

log.trace("A TRACE Message");

log.debug("A DEBUG Message");

log.info("An INFO Message");

log.warn("A WARN Message");

log.error("An ERROR Message");

return "Here are your logs!”; } }

So, there are many such ways in spring boot to use logger.

**What is bootstrapping in spring boot?**

One of the way to bootstrap your spring boot application is using Spring Initializer.

you can go to the official website of spring and select your version, and add you groupID, artifactId and all the required dependencies.

And then you can create your restEndpoints and build and run your project.

There you go, you have bootstrapped your spring boot application.

**How to create jar file in spring boot?**

To create a jar file in spring boot you need to define your packaging file as jar in your pom.xml(if it is maven project).

Then just do maven build with specifying goals as package so that your application will start building.

Once the build is successful, just go into your Target folder and you can see .jar file generated for you application.

**What is dependency injection in spring boot?**

Dependency injection is a way through which the Spring container injects one object into another. This helps for loose coupling of components.

For example: if class student uses functionality of department class, then we say student class has dependency of Department class. Now we need to create object of class Department in your student class so that it can directly use functionalities of department class is called dependency injection.

**How to store image in MongoDB using spring boot?**

One of the way for storing image in MongoDB is by using Spring Content. And also you should have the below dependency in your pom.xml.

<dependency>

<groupId>com.github.paulcwarren</groupId>

<artifactId>spring-content-mongo-boot-starter</artifactId>

<version>0.0.10</version>

</dependency>

You should have a GridFsTemplate bean in your applicationContext.

@Configuration

public class Config

@Bean

public GridFsTemplate gridFsTemplate() throws Exception {

return new GridFsTemplate(mongoDbFactory(), mappingMongoConverter());

}

...

Now add attributes so that your content will be associated to your entity.

@ContentId

private String contentId;

@ContentLength

private long contentLength = 0L;

@MimeType

private String mimeType = "text/plain";

And last but not the least, add a store interface.

@StoreRestResource(path="greatlearningImages")

public interface GreatLearningImageStore extends ContentStore<Candidate, String> {

}

That’s all you have to do to store your images in mongoDb using Springboot.

**How to configure hibernate in spring boot?**

The important and required dependency to configure hibernate is:

**spring-boot-starter-data-jpa**

h2 (you can also use any other database)

Now, provide all the database connection properties in application.properties file of your application in order to connect your JPA code with the database.

Here we will configure H2 database in application.properties file:

spring.datasource.url=jdbc:h2:file:~/test

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=test

spring.datasource.password=test

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

spring.h2.console.enabled=true

spring.h2.console.path=/h2-console

Adding the above properties in your application.properties file will help you to interact with your database using JPA repository interface.

**Explain what is thyme leaf 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 find, springboot automatically configures Thymeleaf to serve dynamic web content.

Dependency: **spring-boot-starter-thymeleaf**

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.

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

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.

Dependency to be added is: spring-boot-devtools

The main focus of this module is to improve the development time while working on Spring Boot applications.

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

Yes, you can change the port of embedded Tomcat server in Spring boot by adding the following property in your application.properties file.

**server.port=8084**

The default port number of the tomcat server to run the spring boot application is 8080, which is further possible to change it.

So we can change the port of tomcat following ways given below:-

Using application.properties

Using application.yml

Using EmbeddedServletContainerCustomizer interface.

Using WebServerFactoryCustomizer interface.

Using Command-Line Parameter.

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

Below are the steps to connect your Spring Boot application to a database using JDBC:

Before that, you need to add required dependencies that are provided by spring-boot to connect your application with JDBC.

**Step 1:** First create a database in MySQL with create DATABASE student;

**Step 2:** Now, create a table inside this DB:

CREATE TABLE student(studentid INT PRIMARY KEY NOT NULL AUTO\_INCREMENT,

studentname VARCHAR(255));

**Step 3**: Create a springBoot and add JDBC,mysql and web dependencies.

**Step 4**: In application.properties, you need to configure the database.

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

spring.datasource.username=system123

spring.datasource.password=system123

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

**Step 5**: In your controller class, you need to handle the requests.

package com.student;

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("/save")

public String index(){

jdbc.execute("insert into student

(name)values(GreatLearnings)"); return "Data Entry Successful"; } }

**Step 6**: Run the application and check the entry in your Database.

**Step 7**: You can also go ahead and open the URL and you will see “Data Entry Successful” as your output.

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

The @RequestMapping annotation can be used at class-level or method level in your controller class.

The global request path that needs to be mapped on a controller class can be done by using @RequestMapping at class-level. If you need to map a particular request specifically to some method level.

Below is a simple example to refer to:

@RestController

@RequestMapping("/greatLearning")

public class GreatLearningController {

@RequestMapping("/")

String greatLearning(){

return "Hello from greatLearning "; }

@RequestMapping("/welcome")

String welcome(){

return "Welcome from GreatLearning"; } }

The @**RestController** annotation is used at the class level.

You can use @RestController when you need to use that class as a request handler class.All the requests can be mapped and handled in this class.

@**RestController** itself consists @Controller and @ResponseBody which helps us to remove the need of annotating every method with @ResponseBody annotation.

Below is a simple example to refer to for use of @RestController annotation:

@RestController

@RequestMapping(“bank-details”)

public class DemoRestController{

@GetMapping(“/{id}”,produces =”application/json”)

public Bank getBankDetails(@PathVariable int id){

return findBankDetailsById(); } }

Here, @**ResponseBody** is not required as the class is annotated with @RestController.

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

AutoConfiguration is a process by which Spring Boot automatically configures all the infrastructural beans. It declares the built-in beans/objects of the spring-specific module such as JPA, spring-security, and so on based on the dependencies present in your application’s classpath.

For example: If we make use of Spring JDBC, the spring boot autoconfiguration feature automatically registers the DataSource and JDBCTemplete bean.

This entire process of automatically declaring the framework-specific bean without the need of writing the XML code or java-config code explicitly is called Autoconfiguration which is done by spring-boot with the help of an annotation called @EnableAutoconfiguration alternatively @SpringBootApplication.

1. You can exclude the attribute of @EnableAutoConfiguration where you don’t want it to be configured implicity in order to disable the spring boot’s auto-configuration feature.

2. Another way of disabling auto-configuration is by using the property file:

**For example:**

**spring.autoconfigure.exclude**=

org.springframework.boot.autoconfigure.mongo.MongoAutoConfiguration,

org.springframework.boot.autoconfigure.data.MongoDataConfiguration,

In the above example, we have disabled the autoconfiguration of MongoDB.

**Can you give an example for ReadOnly as true in Transaction management?**

Yes, example for ReadOnly as true in Transaction Management is:

Suppose you have a scenario where you have to read data from your database like if you have a STUDENT database and you have to read the student details such as studentID, and studentName.

So in such scenarios, you will have to set read-only on the transaction.

**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.

**For example:**

spring: profiles: active: -test

---

spring: profiles: active: -prod

---

spring: profiles: active: -development

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

By using Spring Data Rest, you have access to all the RESTful resources that revolves around Spring Data repositories.

Refer the below example:

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

public interface GreatLearningRepo extends CustomerRepository< greatlearning, Long> { }

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

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.

**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.

@RestController

@RequestMapping("/greatleasrning")

public class Controller {

@Autowired

private final GreatLearningRepository greatLearningRepository;

public Controller(GreatLearningRepository greatLearningRepository) {

this. greatLearningRepository = greatLearningRepository; }

In the above case, your data which may be in JSON format can be inserted successfully into the database.

@RequestMapping(method = RequestMethod.POST)

ResponseEntity<?> insert(@RequestBody Course course) {

greatLearningRepository.save(course);

return ResponseEntity.accepted().build(); } }

**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.

**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.

**How to use crud repository in spring boot?**

In order to use crud repository in spring boot, all you have to do is extend the crud repository which in turn extends the Repository interface as a result you will not need to implement your own methods.

Create a simple spring boot application which includes below dependency:

spring-boot-starter-data-jpa, spring-boot-starter-data-rest

And extend your repository interface as shown below:

package com.greatlearning;

import java.util.List;

import org.springframework.data.repository.CrudRepository;

import org.springframework.data.rest.core.annotation.RepositoryRestResource;

@RepositoryRestResource

public interface GreatLearning extends CrudRepository<Candidate, Long>

{ public List<Candidate> findById(long id);

//@Query("select s from Candidate s where s.age <= ?")

public List<Candidate> findByAgeLessThanEqual (long age); }

**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.

**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.

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;

**What is the rest controller in spring boot?**

The @RestController annotation is used at the class level.

You can use @RestController when you need to use that class as a request handler class.All the requests can be mapped and handled in this class.

@**RestController** itself consists @Controller and @ResponseBody which helps us to remove the need of annotating every method with @ResponseBody annotation.

Below is a simple example to refer to for use of @RestController annotation:

@RestController

@RequestMapping(“bank-details”)

public class DemoRestController{

@GetMapping(“/{id}”,produces =”application/json”)

public Bank getBankDetails(@PathVariable int id){

return findBankDetailsById(); } }

Here, @ResponseBody is not required as the class is annotated with @RestController.

**How to handle 404 error in spring boot?**

Consider a scenario, where there are no stockDetails in the DB and still, whenever you hit the GET method you get 200(OK) even though the resource is not found which is not expected. Instead of 200, you should get 404 error.

So to handle this, you need to create an exception, in the above scenario “StockNotFoundException”.

GetMapping("/stocks/{number}")

public Stock retriveStock(@PathVariable int number)

{

Stock stock = service.findOne(number);

if(Stock ==null)

//runtime exception

throw new StockNotFoundException("number: "+ number);

return stock; }

Now, create a Constructor from Superclass.

Right-click on the file -> Go to Source ->And generate constuctors from superclass-> and check the RuntimeException(String)-> and generate.

And add an annotation called @ResponseStatus which will give you 404 (not found) error.

package com.greatlearning;

import org.springframework.http.HttpStatus;

import org.springframework.web.bind.annotation.ResponseStatus;

@ResponseStatus(HttpStatus.NOT\_FOUND)

public class StockNotFoundException extends RuntimeException

{ public StockNotFoundException(String message)

{ super(message); } }

Now, you can hit the same URL again and there you go, you get a 404 error when a resource is not found.

**Which is the spring boot latest version?**

The latest version of spring boot is 2.6.0. It came out with a lot of dependency upgrades, java 15 support and much more.

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

If we need to set the different target environments, Spring Boot has a built-in mechanism.

One can simply define an application environment.properties file in the src/main/resources directory and then set a Spring profile with the same environment name.

For example, if we define a “production” environment, that means we’ll have to define a production profile and then application-production.properties.

This environment file will be loaded and will take precedence over the default property file. You should note that the default file will still be loaded. It’s just that when there is a property collision, the environment-specific property file takes precedence.

**Where do we define properties in the Spring Boot application?**

**Command Line Properties**

Command-line properties are converted into Spring Boot Environment properties by the spring boot application.

Command-line properties have more precedence over the other property sources.

Spring Boot uses the 8080 port number, by default, to start the Tomcat. Let us see how one can change the port number by using command-line properties.

c:\demo\target>java -jar demo-0.0.1-SNAPSHOT.jar --server.port=9090

**Properties File**

Properties files are used to keep one or more properties in a single file to run the application in a different environment. Properties are kept in the application.properties file under the classpath in a typical spring boot application. The location of the application.properties file is at src/main/resources directory. The code of application.properties file is as below:

* sever.port=9090
* spring.application.name = demoservice

**YAML File**

Spring Boot also supports YAML-based properties configurations to run the application. The user can use, application.yml file instead of the application.properties file. The YAML file is kept inside the classpath. The sample application.yml file is given below −

* spring:
* application:
* name: demoservice
* server:
* port: 9090

**Externalized Properties**

The user can keep properties in different locations or paths instead of keeping the properties file under classpath. While running the JAR file, the user can specify the properties file path. The application developer can use the following command to specify the location of the properties file while running the JAR −

-Dspring.config.location = C:\application.properties

-C:\demo\target>java -jar -Dspring.config.location=C:\application.properties demo-0.0.1-SNAPSHOT.jar

**What is an IOC container?**

IOC Container is a framework that is used for implementing automatic dependency injection. It manages object creation and its lifetime. It, it also injects dependencies into the class.

The IOC container is used to create an object of the specified class. It also injects all the dependency objects through a constructor, a property, or a method at run time and disposes it at the appropriate time. With this, one doesn’t have to create and manage objects manually.

All the containers provide easy support for the Dependency Injection lifecycle as below.

**Register**: The container should know which dependency to instantiate when it encounters a particular type. This process is called registration.

**Resolve**: When using the IoC container, the objects need to be created manually. This is done by the container and is called resolution. The container should include some methods to resolve the specified type; the container creates an object of the specified type. It then injects the required dependencies if any and returns the object.

**Dispose**: The container should manage the lifetime of the dependent objects. IoC containers include different lifetime managers to manage an object’s lifecycle and dispose it.

**What is spring Boot dependency Management?**

Spring Boot manages dependencies and configuration automatically. Each release of spring boot provides a list of dependencies that it supports. The list of dependencies available as a part of Spring-boot dependencies can be used in maven, so we need to specify the version of the dependencies or add the dependencies version in our config file in our configuration.

Spring boot automatically manages and spring boot upgrades all dependencies automatically respectively or consistently at the time when we update the spring boot version.

Advantage of Dependency Management:-

* Spring dependency management provides us the centralized dependency information by adding or specifying the dependencies version in a required place in the spring boot version. It also helps us to switch from one version to another version with ease.
* This management helps us to avoid the mismatch of different versions of the Spring Boot library.
* Here we simply have to write a library name specifying the version.

**Can we create a non-web application in spring boot?**

Yes, but the application could also be called as spring boot standalone application.

To create a non-web application, your application needs to implement CommandLineRunner interface and its Run method for the running of our application. So this run method always acts like the main of our non-web application.

**What is the default port of the tomcat server in Spring Boot?**

As we had already discussed about the default port, the tomcat server in spring boot is port 8080. Which is changeable based on the user or the programmer’s requirement.

**Can we override or replace the embedded tomcat server in spring boot?**

If we consider the fact, spring boot by default comes up with the embedded server once we add the “Spring –boot-starter” dependency. But the spring boot gives us the flexibility to use the tomcat.

If we don’t want to use the tomcat, then tomcat comes with three types of embed servers: Tomcat, jetty, and undertow.

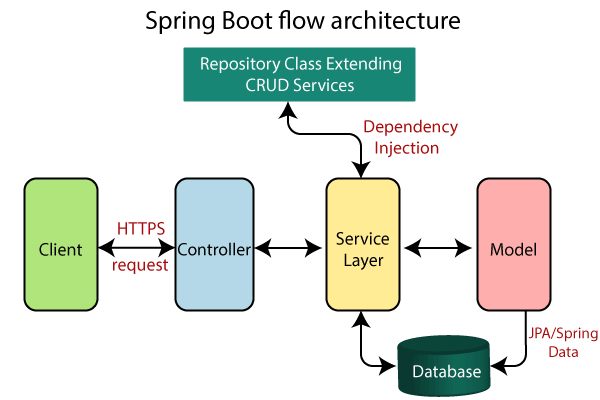
**Can we disable the default web server in the spring boot application?**

Yes, as discussed above, there are 3 web servers available we can choose between them. Spring boot gives more priority for using the tomcat server.

**What is the difference between @RestController and @Controller in Spring Boot?**

**Controller** is a common annotation that is used to mark a class as a spring MVC controller. **Rest controller** is a Springspecial controller used in Restful web services and the wrapped within the @controller and @Responsebody

**Describe the flow of HTTPS request through the spring boot app?**



* We all can see the above image of the spring boot flow architecture to understand the basic concept of the HTTPS request flow in the spring boot app.
* We have the validator classes, view classes, and utility classes.
* As we all know, spring boot uses the modules of spring-like MVC, spring data, etc.
* So the concept also the same for several things, and also the architecture of spring boot is the same as the architecture of spring MVC; instead of one concept, there is no need for the DAO and DAOimpl classes in spring boot.
* It creates a data access layer and started performing CRUD operations.
* CRUD operation is nothing but Create Read Update and Delete operation, which is done by all of the programmers in their website.
* The client makes the HTTP request in PUT or GET.
* After this, the request goes to the controller, and the controller maps that respective request and handles it; if there is the requirement for calling some logic, it calls the service logic after handling the request.
* All the business logic performs in the service layer.
* Service layer performs the logic on the data that is mapped to JPA with model classes.
* A JSP page is returned to the user if no error has occurred.

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

The @GetMapping is a composed annotation which is the short notation of @RequestMapping(method=RequestMethod.GET).

These both methods support the “Consumes.”

The consumes options are,

Consumes=”text/plain”

Consumes={“text/plain”,”application”};

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

In the case of spring boot, you can use appContext.getBeanDefinitionNames() to get all the beans loaded by the spring container.

By calling this method, we can show all of our beans present in our spring boot applications.

**What is Spring Cloud?**

Spring Cloud is an open-source library that provides tools for quickly deploying the JVM based application on the clouds. It provides a better user experience and an extensible mechanism due to various features like Distributed configuration, Circuit breakers, Global locks, Service registrations, Load balancing, Cluster state, Routing, Load Balancing, etc. It is capable of working with spring and different applications in various languages

Features of Spring Cloud

Major features are as below:

* Distributed configuration
* Distributed messaging
* service-to-service calls
* Circuit breakers
* Global locks
* Service registration
* Service Discovery
* Load balancing
* Cluster state
* Routing

**How Do You Override A Spring Boot Project’s Default Properties?**

Spring Application loads properties from the application.properties files in the following locations and add them to the Spring Environment:

A /config subdirectory of the current directory.

The current directory

A classpath /config package

The classpath root

The list is ordered by precedence means that the properties that are defined in locations higher in the list override those defined in lower locations.

If the user does not want application.properties as the configuration file name, they can switch to another by specifying a spring.config.name environment property. The user can also refer to an explicit location using the spring.config.location environment property (comma-separated list of directory locations, or file paths).

$ java -jar myproject.jar --spring.config.name=myproject

or

$ java -jar myproject.jar --spring.config.location=classpath:/default.properties,classpath:/override.properties

**How Is Spring Security Implemented In A Spring Boot Application?**

Spring Security is a framework that majorly focuses on providing both authentication and authorization to Java EE-based enterprise software applications.

Adding Spring security:

Maven: To include spring security, include the below dependency:

<dependencies>

<dependency>

<groupID>org.springframework.security</groupID>

<artifactId>spring-security-config</artifactID>

<version>5.5.0</version>

</dependeny>

<dependency>

<groupId>org.springframework.security</groupId>

<artifactId>spring-security-web</artifactId>

<version>5.5.0</version>

</dependency> </dependencies>

Gradle:

To include spring security in Gradle based project use:

repositories { mavenCentral() }

dependencies {

compile 'org.springframework.security:spring-security-web:5.5.0'

compile 'org.springframework.security:spring-security-config:5.5.0' }

**Which Embedded Containers Are Supported By Spring Boot?**

The embedded containers supported by spring boot are Tomcat (default), Jetty, and undertow servers

**How to Configure Spring Boot Application Logging?**

Spring Boot provides a LoggingSystem abstraction that configures logging based on the content of the classpath. If Logback is available, it is definitely the first choice.

Suppose the only change the user needs to make to logging is to set the levels of various loggers. In that case, they can do so in application.properties by using the “logging.level” prefix, as shown in the following example:

* logging.level.org.springframework.web=DEBUG
* logging.level.org.hibernate=ERROR

**What is the Minimum Java version needed for Spring Boot?**

Java 8 is the minimum version required.

**How to use thymeleaf?**

Steps are as follows:

First, create a Spring Boot Project using STS or Spring Initializer. Add dependency for Thymeleaf and Spring Web.

**For Maven:**

<dependency>

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

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

</dependency>

<dependency>

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

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

</dependency>

Create a Controller Class in package by either adding a new package or use the default package containing the main application class.

**DemoController**.java:

@Controller

public class DemoController {

@GetMapping(value = "/thymeleafTemplate")

public String getTemplate(@RequestParam(name="name" , required=false, defaultValue="World") String name, Model model) {

model.addAttribute("name",name);

return "thymeleafTemplate"; } }

Add template in the resources folder. src/main/resources/templates/**thymeleafTemplate.html**

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

<title> Thymeleaf Spring Boot Demo </title>

</head>

<body>

<p th:text=" 'Hello, ' + ${name} + '!'"/>

<h4> Welcome to Thymeleaf Demo in Spring Boot</h4>

</body> </html>

**Build code.**

Run the application using Integrated Development Environment: Run as -> Spring Boot App.

**How to Use Spring Boot for Command-Line Applications?**

To run Spring Boot for Command-Line Applications, Open the terminal window and change the directory to the root folder of your Spring Boot application.

If the user list files in this directory, they should see a pom.xml file. One can also run your Spring Boot application as an executable Java jar file

**How Can You Change the Default Port in Spring Boot?**

Default port is 8080; The user can change the default port by:

**Command-line:**

java -jar spring-5.jar --server.port=8083

**application.properties**

server.port=8081

**Programmatic Configuration:**

@SpringBootApplication

public class CustomApplication {

public static void main(String[] args {

SpringApplication app = new SpringApplication(CustomApplication.class);

app.setDefaultProperties(Collection.singletonMap("server.port", "8083"));

app.run(args); }

**Explain what happens in the background when a Spring Boot Application is “Run as Java Application”?**

If you are using Eclipse IDE or an Eclipse maven plugin, make sure that as soon as you add a dependency or change the class file, it is compiled and available in the target folder. After that, it is just like any other Java application.

When you launch the java application, then the spring boot auto configuration kicks in. It starts up tomcat when it sees that you are developing a web application!

**What are the differences between JPA and Hibernate?**

JPA is a standard, while Hibernate is not a standard.

The session is used to handle data’s persistence in hibernate, while in JPA, Entity Manager is used. The query language in Hibernate is Hibernate Query language, while in JPA, the query language is Java Persistence query language. Hibernate is one of the most JPA providers.

**What is a spring boot? Why should you use it?**

Spring Boot provides a good platform for Java developers to reduce overall development time and increase efficiency by integrating tests. One can choose Spring Boot because it provides powerful batch processing, eases dependency management, and no manual configurations are needed.

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

The main class in spring boot is configured automatically by the “public static void main()” method that starts up the Spring ApplicationContext.

**What are the spring boot features?**

Some of the important spring boot features are mentioned below:

**Admin support:** Springboot’s admin support feature is used to manage application remotely and also provides the facility to enable admin-related features for the application.

**Externalized Configuration**: Spring Boot’s externalized configuration helps the developers to work with the same application code in a different environment.

**Profiles**: Springboot’s profile feature provides a way to segregate parts of your application and make it be available only in certain environments.

**Logging**: Springboot’s logging feature uses “Commons Logging” for all internal logging.

**Internationalization**: Springboot’s internationalization feature supports localized messages i.e your application can cater to different language preferences.

**JSON:** Spring Boot provides integration of three JSON libraries like Gson, Jackson, JSON-B.

**Web applications:** Spring Boot is one of the platforms that is well suited for web applications.

**Security**: Spring boot is by default secure with basic authentication on all HTTP endpoints.

**How does spring boot handle exception?**

Springboot’s exception handler is an annotation that is used to handle the specific exceptions with the help of @ExceptionHandler annotation.

**How does spring boot Microservice discover dependent Microservices?**

Eureka service can discover dependent microservices in spring boot to get the job done. This service will register all the client microservices through the eureka server to get the dependent microservice.

**What is a bean in spring?**

In Spring, the bean is defined as an object that is like a backbone of your application, managed by a Spring IoC container.

**What is spring boot Microservices?**

Spring Boot microservices enables production-ready applications to iterate fast and provide non-functional features. This is the reason why spring boot microservices has become the de facto standard for Java™. In microservices, you can write code for your single functionality. You can use different technology stacks for different microservices as per the skill set. You can develop this type of microservices with the help of Spring boot very quickly as spring boot gives priority to convention over configuration which increases the productivity of your developers.

**What is the classpath in spring boot?**

Classpath in spring boot is defined as a path where you place resources. During the development, stage maven will take files and place them in the appropriate place for you to use them.

**How does spring boot application work?**

Springboot can configure your application automatically based on the dependencies of the project by using @EnableAutoConfiguration annotation.

Spring vs Spring Boot

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

**What is Spring Boot and mention the need for it?**

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.

So, if you ask me why should anybody use Spring Boot, then I would say, Spring Boot not only improves productivity but also provides a lot of conveniences to write your own business logic.

**Mention the advantages of Spring Boot**

The advantages of Spring Boot are as follows:

* Provides auto-configuration to load a set of default configuration for a quick start of the application
* Creates stand-alone applications with a range of non-functional features that are common to large classes of projects
* It comes with embedded tomcat, servlet containers jetty to avoid the usage of WAR files
* Spring Boot provides an opinionated view to reduce the developer effort and simplify maven configurations
* Provides CLI tool to develop and test applications
* Comes with Spring Boot starters to ensure dependency management and also provides various security metrics
* Consists of a wide range of APIs for monitoring and managing applications in dev and prod.
* Integrates with Spring Ecosystem like Spring JDBC, 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.

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

There is no doubt in the fact that Spring Boot allows the developers to run the same application in different environments. Well, 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. 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. To learn more about Java, it’s recommended to join Java training course today at Edureka.

**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.**

Spring Actuator is a cool feature of Spring Boot with the help of which you can see what is happening inside a running application. So, whenever you want to debug your application, and need to analyze the logs you need to understand what is happening in the application right? In such a scenario, the Spring Actuator provides easy access to features such as identifying beans, CPU usage, etc. The Spring Actuator provides a very easy way to access the production-ready REST points and fetch all kinds of information from the web. These points are secured using Spring Security’s content negotiation strategy.

**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 that dependencies.

**Mention the minimum requirements for a Spring boot System?**

Spring Boot Logo - Spring Boot Interview Questions - EdurekaSpring 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 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?**

Yes, we can change the port of the embedded tomcat server by using the application properties file. In this file, you have to add a property of “server.port” and assign it to any port you wish to. For example, if you want to assign it to 8081, then you have to mention server.port=8081. Once you mention the port number, the application properties file will be automatically loaded by Spring Boot and the required configurations will be applied on to the application.

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

Spring Boot Dev Tools are an elaborated set of tools and aims to make the process of developing an application easier. If the application runs in the production, then this module is automatically disabled, repackaging of archives are also excluded by default. So, the Spring Boot Developer Tools applies properties to the respective development environments. To include the DevTools, you just have to add the following dependency into the pom.xml file:

<dependency>

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

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

</dependency>

**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 following steps need to be followed to create a Spring Boot project using Spring Initializer:

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.

Once, all the above steps are followed you will see that the Spring Boot project is created with all the required dependencies.

**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 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:

@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.

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

1)@RequestMapping 2)@RestController

1) This annotation is used to provide the routing information and tells to Spring that any HTTP request must be mapped to the respective method.

2) This annotation is used to add the @ResponseBody and @Controller annotation to the class

1) To use this annotation, you have to import org.springframework.web.bind.annotation. RequestMapping;

2) To use this annotation, you have to import org.springframework.web.bind.annotation. RestController;

**Mention the differences between JPA and Hibernate.**

1)JPA is a Data Access Abstraction used to reduce the amount of boilerplate code

2)Hibernate is an implementation of Java Persistence API and offers benefits of loose coupling

**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, Jersey, etc.

**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. 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**

1)Used in the main class or bootstrap class

1)Used to enable auto-configuration and component scanning in your project

2)It is a combination of @Configuration, @ComponentScan and @EnableAutoConfiguration annotations.

2)It is a combination of @Configuration and @ComponentScan annotations

**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:

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>**

**Can you give an example for ReadOnly as true in Transaction management?**

Example for ReadOnly as TRUE in transaction management could be as follows:

Consider a scenario, where you have to read data from the database. For example, let us say you have a customer database, and you want to read the customer details such as customerID, and customername. To do that, you will set read-only on the transaction as we do not want to check for the changes in the entities.

**Can you explain how to deploy to a different server with Spring Boot?**

To deploy a different server with Spring Boot, follow the below steps:

**What is Spring Boot and what are its Benefits?**

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.

**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.

1. Integrates a tool called the Actuator, which enables users to manage and monitor applications
2. Provides support for embedded servers, such as Jetty and Tomcat
3. Users can simply run war files, without deploying it
4. Includes an Auto-Configuration functionality, allowing users to configure Spring applications automatically
5. 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:

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

**What are the key dependencies of Spring Boot?**

Mentioned below are important Spring Boot dependencies that need to be added to a Gradle-based or Maven-based application, to ensure application compatibility with Spring Boot features.

* spring-boot-starter-parent
* spring-boot-maven-plugin
* spring-boot-starter-test
* spring-boot-starter-security
* spring-boot-starter-actuator
* Spring-boot-starter-web

**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 "); }}

**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.

**What are Profiles in Spring Boot?**

Profiles in the Spring framework enables users to map components and beans to specific profiles, such as the Development (dev) profile, Production (prod) profile, or the Test profile.

In Spring Boot, the annotation @Profile is used to map components and beans to a certain profile.

Developers can also set up profiles using the SpringApplication, for instance, SpringApplication.setAdditionalProfiles("dev");

**What differentiates Spring Data JPA and Hibernate?**

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.

**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.

**What Is Spring Boot and What Are Its Main Features?**

Spring Boot is essentially a framework for rapid application development built on top of the Spring Framework. With its auto-configuration and embedded application server support, combined with the extensive documentation and community support it enjoys, Spring Boot is one of the most popular technologies in the Java ecosystem as of date.

**Here are a few salient features:**

1. Starters – a set of dependency descriptors to include relevant dependencies at a go
2. Auto-configuration – a way to automatically configure an application based on the dependencies present on the classpath
3. Actuator – to get production-ready features such as monitoring
4. Security
5. Logging

**What Are the Differences Between Spring and Spring Boot?**

The Spring Framework provides multiple features that make the development of web applications easier. These features include dependency injection, data binding, aspect-oriented programming, data access and many more.

Over the years, Spring has been growing more and more complex, and the amount of configuration such application requires can be intimidating. This is where Spring Boot comes in handy — it makes configuring a Spring application a breeze.

Essentially, while Spring is unopinionated, Spring Boot takes an opinionated view of the platform and libraries, letting us get started quickly.

Here are two of the most important benefits Spring Boot brings in:

Auto-configure applications based on the artifacts it finds on the classpath

Provide non-functional features common to applications in production, such as security or health checks

**What is Spring Boot?**

Over the years spring has become more and more complex as new functionalities have been added. Just visit the page-https://spring.io/projects and we will see all the spring projects we can use in our application for different functionalities. If one has to start a new spring project we have to add build path or add maven dependencies, configure application server, add spring configuration . So a lot of effort is required to start a new spring project as we have to currently do everything from scratch. Spring Boot is the solution to this problem. Spring boot has been built on top of existing spring framework. Using spring boot we avoid all the boilerplate code and configurations that we had to do previously. Spring boot thus helps us use the existing Spring functionalities more robustly and with minimum efforts.

**What are advantages of Spring Boot?**

The advantages of Spring Boot are

1. Reduce Developement, Testing time and efforts.
2. Use of JavaConfig helps avoid usage of XML.
3. Avoid lots of maven imports and the various version conflicts.
4. Provide Opinionated Development approach.
5. Quick start to development by providing defaults.
6. No Separate Web Server Needed.Which means that you no longer have to boot up Tomcat, Glassfish, or anything else.
7. Requires less configuration-Since there is no web.xml file. Simply add classes annotated with@Configuration and then you can add methods annotated with@Bean, and Spring will automagically load up the object and manage it like it always has. You can even add @Autowired to the bean method to have Spring autowire in dependencies needed for the bean.
8. Environment Based Configuration-Using these properties, you can pass into the application which environment you are using with:-Dspring.profiles.active={enviornment}. Spring will then load up the subsequent application properties file at (application-{environment}.properties) after loading up the main application properties file.

**Which build tool have you used to develop Spring Boot Application?**

Spring Boot application can be developed using Maven as well as Gradle.

Spring Boot application using Maven

Spring Boot application using Gradle

**What is JavaConfig?**

Spring JavaConfig is a product of the Spring community that provides a pure-Java approach to configuring the Spring IoC Container. It thus helps avoid using XML configurations. The advantages of using JavaConfig are

The advantages of JavaConfig are

* Object-oriented configuration. Because configurations are defined as classes in JavaConfig, users can take full advantage of object-oriented features in Java. One configuration class may subclass another, overriding its @Bean methods, etc.
* Reduced or eliminated XML configuration. The benefits of externalized configuration based on the principles of dependency injection have been proven. However, many developers would prefer not to switch back and forth between XML and Java. JavaConfig provides developers with a pure-Java approach to configuring the Spring container that is conceptually similar to XML configuration. It is technically possible to configure the container using only JavaConfig configuration classes, however in practice many have found it ideal to mix-and-match JavaConfig with XML.
* Type-safe and refactoring-friendly. JavaConfig provides a type-safe approach to configuring the Spring container. Thanks to Java 5.0's support for generics, it is now possible to retrieve beans by type rather than by name, free of any casting or string-based lookups.

**How to reload my changes on Spring Boot without having to restart server?**

This can be achieved using DEV Tools. With this dependency any changes you save, the embedded tomcat will restart. Spring Boot has a Developer tools (DevTools) module which helps to improve the productivity of developers. One of the key challenge for the Java developers is to auto deploy the file changes to server and auto restart the server. Developers can reload changes on Spring Boot without having to restart my server. This will eliminates the need for manually deploying the changes every time. Spring Boot doesnt have this feature when it has released its first version. This was a most requested features for the developers. The module DevTools does exactly what is needed for the developers. This module will be disabled in the production environment. It also provides H2-database console for better testing the application. The following dependency is used

<dependency>

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

<artifactId>spring-boot-devtools</artifactId> <optional>true</optional>

</dependency>

**What is Actuator in Spring Boot?**

Spring boot actuator is one of the important feature in spring boot framework. Spring boot actuator helps you to access the current state of the running application in production environment. There are several metrics that has to be checked and monitored in the production environment. Even some external applications may be using those services to trigger the alert message to concerned person. Actuator module exposes set of REST endpoints that can be directly accessed as a HTTP URL to check the status.

**How to implement JWT authentication for Spring Boot Application?**

We will be implementing Spring Boot +JWT + MYSQL Hello World Example

JWT stands for JSON Web Token. JSON Web Token (JWT) is an open standard (RFC 7519) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object. This information can be verified and trusted because it is digitally signed. The client will need to authenticate with the server using the credentials only once. During this time the server validates the credentials and returns the client a JSON Web Token(JWT). For all future requests the client can authenticate itself to the server using this JSON Web Token(JWT) and so does not need to send the credentials like username and password.

**How to deploy Spring Boot application as a WAR?**

Spring Boot WAR Deployment

**How to disable Actuator endpoint security in Spring Boot?**

By default all sensitive HTTP endpoints are secured such that only users that have an ACTUATOR role may access them. Security is enforced using the standard HttpServletRequest.isUserInRole method.

We can disable security using -

management.security.enabled=false

It is suggested to disable security only if the actuator endpoints are accessed behind firewall.

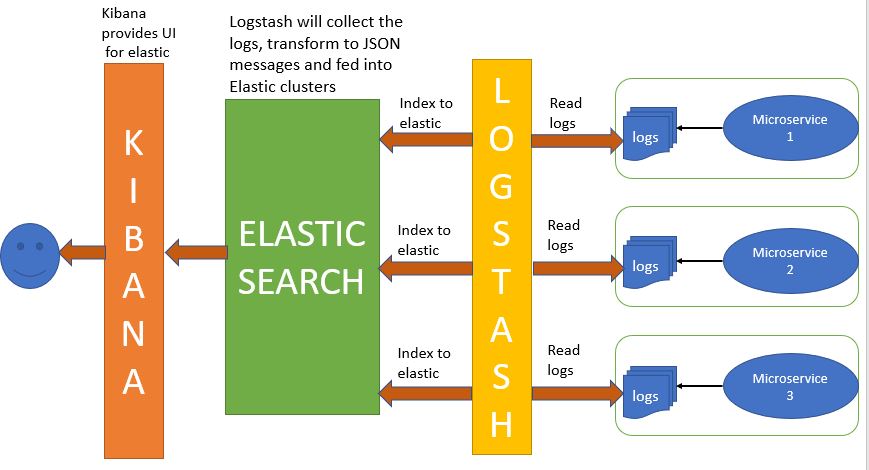
**How to run Spring boot application to custom port?**

In order to run a spring boot application on a custom port you can specify the port in application.properties.

server.port=8090

**What is ELK stack?How to use it with Spring Boot?**

The ELK Stack consists of three open-source products - Elasticsearch, Logstash, and Kibana from Elastic.



Elasticsearch is a NoSQL database that is based on the Lucene search engine.

Logstash is a log pipeline tool that accepts inputs from various sources, executes different transformations, and exports the data to various targets. It is a dynamic data collection pipeline with an extensible plugin ecosystem and strong Elasticsearch synergy

Kibana is a visualization UI layer that works on top of Elasticsearch.

These three projects are used together for log analysis in various environments. So Logstash collects and parses logs, Elastic search indexes and store this information while Kibana provides a UI layer that provide actionable insights.

Spring Boot + ELK stack

Have you written Test cases using Spring Boot?

Spring Boot provides the @SpringBootTest for writing Unit Test Cases

Spring Boot Unit Test Simple Example

**What is YAML?**

YAML is a human-readable data serialization language. It is commonly used for configuration files.

Compared to properties file, YAML file is much more structured and less confusing in case we want to add complex properties in the configuration file. As can be seen YAML has hierarchical configuration data.

**How to implement security for Spring boot application?**

For Implementing security for Spring Boot we use the spring-boot-starter-security dependency and have to add the Security config. It requires very little code. Config class will have to extend WebSecurityConfigurerAdapter and override its methods.

**Have you integrated Spring Boot and ActiveMQ?**

For integrating Spring Boot and ActiveMQ we use the spring-boot-starter-activemq dependency

It requires very little configuration and no boilerplate code.

**How to implement Pagination and Sorting with Spring Boot?**

Using Spring Boot achieving pagination is very simple. Using the Spring Data-JPA this is achieved passing pageable org.springframework.data.domain.Pageable to the repository methods.

**What is Swagger? Have you implemented it using Spring Boot?**

Swagger is widely used for visualizing APIs, and with Swagger UI it provides online sandbox for frontend developers. For the tutorial, we will use the Springfox implementation of the Swagger 2 specification. Swagger is a tool, a specification and a complete framework implementation for producing the visual representation of RESTful Web Services. It enables documentation to be updated at the same pace as the server. When properly defined via Swagger, a consumer can understand and interact with the remote service with a minimal amount of implementation logic. Thus Swagger removes the guesswork in calling the service.

**What is Spring Profiles? How do you implement it using Spring Boot?**

Spring Profiles allows users to register beans depending on the profile(dev, test, prod etc). So when the application is running in DEVELOPMENT only certain beans can be loaded and when in PRODUCTION certain other beans can be loaded. Suppose our requirement is that the Swagger documentation be enabled only for the QA environment and disabled for all others. This can be done using Profiles. Spring Boot makes using Profiles very easy.

**What is Spring Batch? How do you implement it using Spring Boot?**

Spring Boot Batch provides reusable functions that are essential in processing large volumes of records, including logging/tracing, transaction management, job processing statistics, job restart, skip, and resource management. It also provides more advanced technical services and features that will enable extremely high-volume and high performance batch jobs though optimization and partitioning techniques.Simple as well as complex, high-volume batch jobs can leverage the framework in a highly scalable manner to process significant volumes of information.

**What is FreeMarker Template? How do you implement it using Spring Boot?**

FreeMarker is a Java-based Template Engine, originally focusing on dynamic web page generation with MVC software architecture. The major advantage of using Freemarker is the complete separation of the Presentation layer and the Business Layer. The Programmers can work on the application code while the designers can work on the html page design. Finally using freemarker these can then be combined to give the final output page.

**How to implement Exception Handling using Spring Boot?**

Spring provides a very useful way to handle exceptions using ControllerAdvice.

We will be implementing a ControlerAdvice class which will handle all exceptions thrown by the controller class.

**What is caching? Have you used any caching framework with Spring Boot?**

A cache is an area of local memory that holds a copy of frequently accessed data that is otherwise expensive to get or compute. Have used Hazelcast for caching.

**Have you exposed a SOAP webservice endpoint using Spring Boot?**

Yes. Using Spring Boot exposed a web service to be consumed. Used Contract first approach to generate the classes from wsdl.

**How did you perform database operations using Spring Boot?**

Spring Boot Tutorial-Spring Data JPA

Spring Boot JDBC Example

**How to develop a full stack application using Spring Boot and Angular?**

In full stack application we expose the back end point to get the data. This data can then be used by any application or device as per the need. In future even if another front end device is to be used, there will not be much change and the new device will need to consume these end points.

**Which all starter maven dependencies have you used?**

Have used different starter dependencies like spring-boot-starter-activemq dependency, spring-boot-starter-security dependency, spring-boot-starter-web dependency.

This helps in adding less number of dependencies and also in reducing version conficts.

**What is CSRF attack? How to enable CSRF protection against it?**

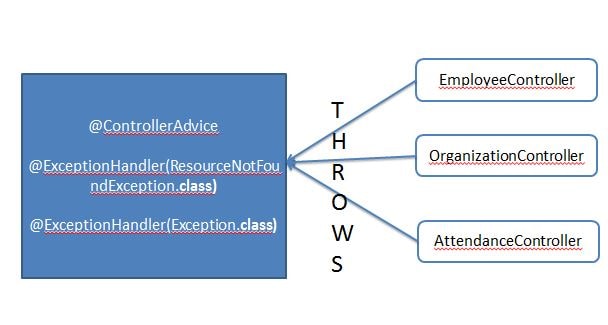
CSRF stands for Cross-Site Request Forgery. It is an attack that forces an end user to execute unwanted actions on a web application in which they are currently authenticated. CSRF attacks specifically target state-changing requests, not theft of data, since the attacker has no way to see the response to the forged request.

**Have you used any integration framework with Spring Boot?**

Have integrated Apache Camel with Spring Boot. Made use of Apache Camel Spring Boot starter dependency. Spring Boot +Apache Camel

**When will you use WebSockets? How to implement it using Spring Boot?**

WebSocket is a computer communications protocol, providing full-duplex communication channels over a single TCP connection



WebSocket are bi-directional - Using WebSocket either client or server can initiate sending a message.

WebSocket are Full Duplex - The client and server communication is independent of each other.

Single TCP connection - The initial connection is using HTTP, then this connection gets upgraded to a socket based connection. This single connection is then used for all the future communication

Light - The WebSocket message data exchange is much lighter compared to http.

**What is AOP? How to use it with Spring Boot?**

During software development, functions that span multiple points of an application are called cross-cutting concerns. These cross-cutting concerns differ from the application’s main business logic. Hence ,separating these cross-cutting concerns from the business logic is where aspect-oriented programming (AOP) comes into picture.

**What is Apache Kafka? How to integrate it with Spring Boot?**

Apache Kafka is a distributed publish-subscribe messaging system. It is a scalable, fault-tolerant, publish-subscribe messaging system which enables us to build distributed applications. It is an Apache Top Level project. Kafka is suitable for both offline and online message consumption.

**How can we monitor all the Spring Boot Microservices?**

Spring Boot provides actuator endpoints to monitor metrics of individual microservices. These endpoints are very helpful for getting information about applications like if they are up, if their components like database etc are working good. But a major drawback or difficulty about using actuator enpoints is that we have to individually hit the enpoints for applications to know their status or health. Imagine microservices involving 50 applications, the admin will have to hit the actuator endpoints of all 50 applications. To help us deal with this situation, we will be using open source project located at https://github.com/codecentric/spring-boot-admin.

Built on top of Spring Boot Actuator, it provides a web UI to enable us visualize the metrics of multiple applications.

**Have you used any Spring Cloud Components with Spring Boot?**

Have used Spring Cloud components like Netflix Eureka for Service Registration,Ribbon for Load Balancing.

**How to deploy Spring Boot Application to Pivotal Cloud Foundry(PCF)?**

Deploying Spring Boot Application to PCF

**What is JWT ? How to implement it using Spring Boot Security?**

For better understanding we will be developing the project in stages

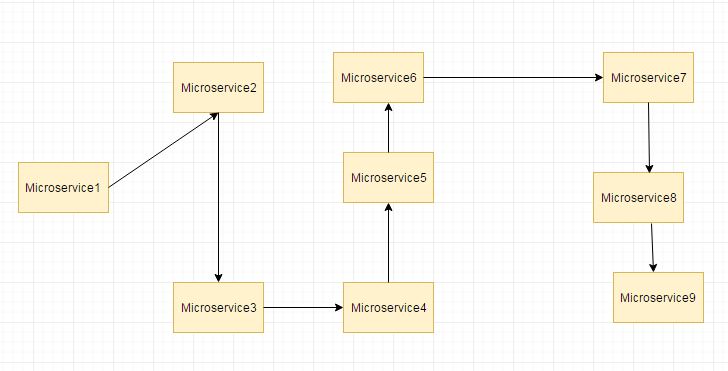
Develop a Spring Boot Application to expose a Simple REST GET API with mapping /hello.

Configure Spring Security for JWT. Expose REST POST API with mapping /authenticate using which User will get a valid JSON Web Token. And then allow the user access to the api /hello only if it has a valid token

**How to implement distributed logging for microservices?**

Microservices architecture involve multiple services which interact with each other. So a functionality may involve call to multiple microservices. Usually for systems developed using Microservices architecture, there are many microservices involved. These microservices collaborate with each other.

Consider the following microservices-



If suppose during such calls there are some issues like exception has occurred. Or may be there are latency issues due to a particular service taking more than expected time. How do we identify where the issue is occurring. In regular project we would have used logging to analyze the logs to know more about occurred exceptions and also performance timing. But since microservices involves multiple services we cannot use regular logging. Each Service will be having its own separate logs. So we will need to go through the logs of each service. Also how do we correlate the logs to a request call chain i.e which logs of microservices are related to Request1, which are related to Request2. To resolve these issues we make use of Spring Cloud Sleuth and Zipkin

Spring Cloud Sleuth is used to generate and attach the trace id, span id to the logs so that these can then be used by tools like Zipkin and ELK for storage and analysis

Zipkin is a distributed tracing system. It helps gather timing data needed to troubleshoot latency problems in service architectures. Features include both the collection and lookup of this data.

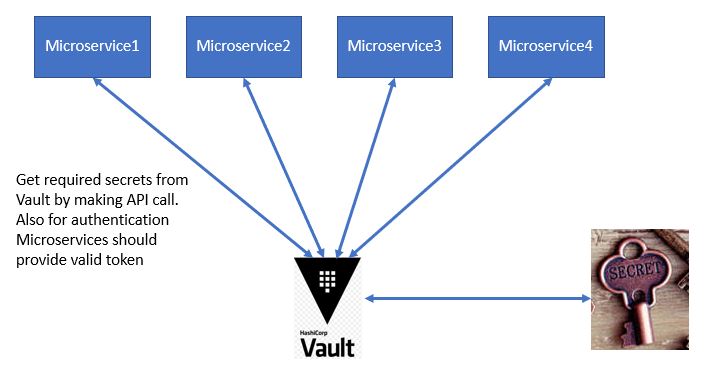
**What is Hashicorp Valut? How to use it with microservices?**

Microservices architecture have multiple services which interact with each other and external resources like databases. They also need access to usernames and passwords to access these resources. Usually these credentials are stored in config properties. So each microservice will have its own copy of credentials. If any credentials change we will need to update the configurations in all microservices. We have previously discussed one solution to this problem is using Spring Cloud Config Native Server or Spring Cloud Config Git Server where common global properties which are repeated in all the microservices are usually stored. But still storing the secrets in configuration file is a security concern. Above approach as 2 drawbacks-

No single point of Truth

Security risk of exposing the credentials

In this tutorial will be using Spring Cloud Config and Hashicorp Vault to manage secrets and protect sensitive data.



Hashicorp Vault is a platform to secure, store, and tightly control access to tokens, passwords, certificates, encryption keys for protecting sensitive data and other secrets in a dynamic infrastructure.

Using vault we will be retrieving the credentials from the vault key/value store.

**Explain the term ‘Spring Boot’.**

It is a Spring module that offers Rapid Application Development to Spring framework. Spring module is used to create an application based on Spring framework which requires to configure few Spring files.

**Mention some advantages of Spring Boot**

Here are some major advantages of using spring-boot:

Helps you to create a stand-alone application, which can be started using java.jar.

It offers pinpointed‘started’ POMs to Maven configuration.

Allows you to Embed Undertow, Tomcat, or Jetty directly.

Helps you to configure spring whenever possible automatically.

**How to create a Spring Boot application using Spring Initializer?**

It is a web tool provided by Spring on its official website. However, you can also create Spring Boot project by entering project details.

* Starter dependency
* Auto-configuration
* Spring initializer

**Explain different phases of RAD model.**

This is a frequently asked job interview. Various phases of RAD mode are:

Business Modeling: Based on the flow of information and distribution between various business channels, the product is designed.

Data Modeling : The information collected from business modeling is refined into a set of data objects that are significant for the business.

Application Generation: Automated tools are used for the construction of the software, to convert process and data models into prototypes.

**What is RAD model?**

RAD or Rapid Application Development process is an adoption of the waterfall model; it targets developing software in a short period. RAD follow the iterative

SDLC RAD model has the following phases:

* Business Modeling
* Data Modeling
* Process Modeling
* Application Generation
* Testing and Turnover

**What are the commands to run and stop Spring Boot executable jar file?**

You need to open cmd or shell window command and use

java -jar

Example

$ java -jar myproject-0.0.1-SNAPSHOT.jar

To stop use ctrl+C

**How can you change JDK version in Spring Boot?**

To change the JDK version in Spring Boot, you can overwrite it by adding a java. version property tag as given:1.8

**What is the process that you need to follow to run Spring Boot application on the custom port?**

In order to run a Spring Boot application, you require to put server.port properties in application.properties. For example, server.port=8050

**What is Spring Boot starter? How is it useful?**

Spring Boot has many starters. They are a set of convenient dependency descriptors. Starter allows you to include these descriptors in your pom.xml.

**For example,** If you want to work with Spring MVC, you can include “spring–boot–starter–web” as a dependency in pom.xml.

**Can you use Spring Boot with applications which are not using Spring?**

No, it is not possible as Spring Boot is limited to Spring application only.

**What is the name of the configuration file which you can use in Spring Boot?**

The configuration file used in Spring Boot projects is called application.properties. It is an important file which allows you to override your default configurations.

**What is DevTools in Spring Boot?**

Spring Boot DevTools helps you to increase the productivity of the developer. So, you don’t require to redeploy your application every time you make the changes. It allows the developer to reload changes without the need of restarting of the server.

**What are the important features of Spring Boot?**

Important features of Spring Boot are:

1. Web Development
2. Spring Application
3. Application occasions and listeners
4. Admin highlights
5. YAML Support
6. Type-safe Configuration
7. Externalized Configuration
8. Properties Files
9. Logging and Security

**What are the essential components of Spring Boot**

The important components of Spring Boot are:

* Spring Boot Starter
* Spring Boot autoconfiguration
* Spring Boot Actuator
* Spring Boot CLI

**How are properties defined? Where?**

You can define properties in the application.properties file exists in the classpath.

Example: configure default DataSource bean

database.host=localhost

**What is spring-boot-starter-parent?**

It is a special starter which makes Gradle or Maven dependency-management easy by adding jars to your classpath.

**How to enable HTTP/2 supports in Spring Boot?**

User can enable HTTP/2 support by using

server.http2.enabled configuration property.

**What is a Spring Boot Actuator?**

Spring Boot Actuator allows you to monitor and manage your application when you want to push it for the production. It helps you to control your application by using HTTP endpoints.

**What is the command to run Spring Boot application to custom port?**

In application.properties, add following property.

server.port = 8181

**21) How can you access a value defined in the application? What is properties file in Spring Boot?**

Use the @Value annotation to access the properties which is defined in the application – properties file.

@Value("${custom.value}")

private String customVal;

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

Spring is a web application development framework based on Java. On the other hand Spring Boot is an extension of the spring framework which eliminated the boilerplate configuration required for setup a Spring application.

**Explain Spring Boot Admin**

Spring Boot admin is a community project which helps you to manage and monitor your Spring Boot applications.

**How can you connect Spring Boot to the database using JPA?**

Spring Boot supports spring-boot-data-JPA start, which helps you to connect spring application with a relational database.

**Explain @RestController annotation in Spring Boot?**

The @RestController annotation helps you to add @ResponseBody and @Controller annotations to the class.

You can also import org.springframework.web.bind.annotation package in your file.

**Define the term Spring Initializer**

Spring initializer is a web application which can create an initial project structure for you.

**Explain Spring CLI**

Spring CLI is used for writing in Groovy Spring Boot application, which helps you to concise code.

**Where can you define properties in Spring Boot application?**

You can define properties of Spring Boot into a file called application.properties. It helps you to create this file manually, or you can use Spring Initializer to create this file.

**What are embedded containers support by Spring**

Spring Boot support the main three embedded containers:

1) Tomcat

2) Jetty

3) Undertow.

By default, it uses Tomcat as an embedded container.

**Explain thymeleaf in Spring Boot**

Thymelaf is a server-side Java template engine for a web application. It helps you to bring elegant natural templates to your web application.

**What are the Spring Boot properties?**

Spring Boot offers various properties which can be specified inside our project’s application.properties file. It helps you to set values like a server-port number, database connection configuration, etc.

**What is the main difference between JPA and Hibernate?**

The main difference between both of them is that JPA is a specification/Interface, whereas Hibernate is only JPA implementations.

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

A shutdown is an endpoint that helps application to be shut down properly. This feature is not enabled by default.

However, you can use it by setting command: management.endpoint.shutdown.enabled=true in your application.properties file.

**Is it possible to replace or override the Embedded Tomcat server in Spring Boot?**

Yes, it is possible to replace the Embedded Tomcat with any other servers by using the starter dependencies. For that, you can use spring-boot-starter-jetty or as a dependency for according you to your need.

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

Yes, we can disable the default web server by using application.properties to configure the web application type.

**How do you Add, Filter to an application?**

There are three methods to add filter to Spring Boot application:

By implementing Filter interface.

* Using FilterRegistrationBean.
* Using MVC controller.

**What are Spring Boot Starter Projects?**

Starters in Spring Boot are a set of convenient descriptors that are included in Spring Boot applications. It comes with a variety of Spring-related technology which makes the entire process of the application development much easier.

**What is @pathVariable?**

@PathVariable annotation helps you to extract information from the URI directly.

**What is Swagger2?**

Swagger is used to describing the structure of APIs. Swagger 2 is an open-source service provided in Spring Boot which makes it easier for the machines to find out the structure of APIs like RESTful Web services.

**What are different environments for enterprise application development?**

1. Dev
2. QA
3. Stage
4. Production+

**What are the major differences between RequestMapping and GetMapping?**

RequestMapping can be used with GET, POST, PUT, and many other request methods using the method attribute on the annotation. Whereas GetMapping is only an extension of RequestMapping, which helps you to improve clarity on requests.

**How can you define properties in Spring Boot?**

You can define properties in Spring Boot with the help of the application.properties file which exists in a classpath of the application as follows.

**How to create a Spring Boot project using Maven?**

Use any of the following methods to create a project.

1. Spring Initializr
2. Spring Boot CLI
3. Spring Starter Project Wizard

**What is the use of profiles in Spring Boot?**

Profiles are used to separate various parts of your spring application configuration and make it only available in certain environments.

**How to change tomcat HTTP port?**

To change the tomcat HTTP port, you have to change default HTTP property in application.properties file.

**What is LiveReload in Spring Boot?**

LiveReload is a spring-boot-devtools module that includes LiveReload server to trigger a browser refresh when a resource is changed. LiveReload server extensions are available freeware for Firefox, Chrome, and Safari.

**What are the major benefits of spring Externalized Configuration?**

Externalized Configuration helps to work with the same code in different environments. Developers can use YAML files, properties files, command-line arguments, and environment variables to externalize configuration.

**What do you mean by hot-swapping in Spring Boot?**

It is a way to reload the changes without restarting Tomcat, or Jetty server. Eclipse and Many other IDEs support bytecode hot swapping. If you make any changes that don’t affect the method signature, it should reload without side effect.

**Explain Auto-Configuration in Spring Boot.**

Auto-configuration is used to configure Spring application automatically based on dependencies of classpath parameter. It makes development faster and easier.

**What is the meaning of Aspect-Oriented Programming (AOP)?**

Aspect-Oriented Programming supplements Object-Oriented Programming that aims to increase modularity. AOP breaks program logic into various parts, which are called concerns.

**How to enable logging in Spring Boot?**

In order to enable debug logging, you can specify –debug while starting the application from the command prompt.

**Explain overriding default properties in Spring Boot application.**

Spring Boot has lots of properties that can be easily overridden by specifying them in application.properties.

**Explain Docker in Spring Boot.**

It is a tool designed to create, deploy, and run a project by using containers.

**Define ELK stack.**

The ELK Stack is made of three open-source products: 1) Elasticsearch, 2) Logstash, and 3) Kibana.

**Elasticsearch**: It is a NoSQL database which is based on the open-source search engine called Lucene.

**Logstash**: It is a data processing pipeline tool which accepts inputs from sources, performs different transformations, and exports the data to targets.

**Kibana**: Kibana helps users to visualize data with graphs and chart in Elasticsearch.

**How to handle exception in Spring Boot.**

Spring Boot provides a very useful way to handle exceptions using @ControllerAdvice annotation.

**Explain caching.**

Caching is a memory are that temporary stores frequently accessed data that is otherwise expensive to get or compute.

**What is Cross-Site Request Forgery attack?**

Cross-Site Request Forgery attack or one-click attack is an attack that forces other users to execute malicious commands on the application. CSRF attack specifically targets state-changing requests.

**Define apache freemarker.**

Freemarker is a Java-based template used to generate plain text, emails, HTML file, etc.

**What is mean by spring batch?**

Spring Boot Batch provides code reusability which is important when working with large numbers of records, including transaction management, logging, skipping, job processing statistics, and job restarts.

**Explain Apache Kafka.**

Apache Kafka is an open-source messaging platform. LinkedIn develops it. Apache Kafka enables the user to build distributed applications and handle real-time data feeds. Kafka is suitable for both offline and online messaging.

**Explain CORS in Spring Boot?**

CORS stands for Cross-Origin Resource Sharing is a mechanism implemented by browsers and helps users to authorize cross-domain requests. This mechanism serves as an alternative to less secure and less powerful hacks of the kinds of IFrame or JSONP.

**Explain different types of dependency injection.**

There are two types of dependency injection in Spring Boot. They are as follows:

**Constructor based dependency injection**: It is a technique in which one class object supplies the dependency of another object.

**Setter-based dependency injection**: It is a dependency injection in which the framework injects the primitive and string-based values using setter method.

**What are the advantages of micro service?**

Following are the major advantages of micro service:

1. It makes development fast and easy.
2. Compatible with all container.
3. Reduce production time.
4. It’s a lightweight model that supports a major business application.

**What is the default package in Spring Boot?**

A class without any package declaration is considered as a default package.

**Explain the difference between an embedded container and a WAR.**

The main difference between these two is:

Embedded containers help you to run Spring Boot application as a JAR from the command prompt without setting up any web server, while to run a WAR you need first to set up Tomcat.

**Explain Spring MVC.**

It is a traditional web application framework which helps you to build a web application. This framework is similar to the framework of Struts.

What is the use of <set> tag?

This tag is used to write to inject java set using XML.

**What do you mean by aspect?**

It is a set of APIs which provides cross-cutting requirements.

**What is join point in Spring Boot?**

It is a program execution point like the handling of an exception or the execution of a method. In AOP, a join point is referred to as a method execution.

**How can you set active profile in Spring Boot?**

Follow the following methods to set an active profile in Spring Boot.

Pass this profile as an argument when you launch the Spring Boot application.

Set active the active profile in application.properties file.

**Is excluding package without using the basePackages filter is possible? How?**

Yes. It is possible to exclude package without using the basePackages filter by simply using the exclude attribute while using the @SpringBootApplication annotation.

**List out benefits of using the JavaConfig method.**

Following are the benefits of JavaConfig method.

User can take benefit of object-oriented configuration.

Spring Boot configuration improves the efficiency of web-based application by eliminating complex XML configuration.

**Explain steps to deploy an application on virtual machine.**

Below are the steps to deploy application on virtual machine.

1. Install Java.
2. Install the Application Server.
3. Deploy the application war file.

**List out some of the Spring Boot Starters.**

Different Spring Boot Starters are as follows:

1. Security
2. Parent
3. web
4. Thymeleaf
5. Freemarker