**Day 17 Assignment SpringBoot:**

**14. What do you understand by the term ‘Spring Boot’?**

Spring Boot is an open-source Java-based framework that is used for building and deploying standalone, production-grade applications with ease. It provides a comprehensive infrastructure support to develop and deploy microservices-based applications quickly and efficiently. Spring Boot is built on top of the popular Spring Framework and allows developers to create stand-alone, production-grade applications that can be started with minimal configuration and setup.

With Spring Boot, developers can quickly create web applications, RESTful services, and other types of applications using a simple, opinionated approach to configuration and setup. Spring Boot takes care of much of the low-level infrastructure and boilerplate code required to set up a project, making it much easier and faster to get started with Spring development.

**15.Explain the advantages of using Spring Boot for application development.**

There are several advantages of using Spring Boot for application development:

1. Rapid Development: Spring Boot provides a range of features that simplify the development process and speed up application development. The built-in auto-configuration and starter dependencies enable developers to create an application quickly with minimal setup.
2. Easy to Learn: Spring Boot is easy to learn and use for developers who are familiar with the Java programming language and the Spring Framework. Its opinionated approach to configuration and simplified development process makes it a popular choice for building applications.
3. Microservices Support: Spring Boot has built-in support for developing microservices-based applications. It includes features such as embedded servers, service discovery, and distributed tracing, which make it easy to build scalable and resilient microservices.
4. Reduced Boilerplate Code: Spring Boot reduces the amount of boilerplate code required to configure an application, making it easier to focus on business logic. The use of annotations and a declarative programming model further simplifies the development process.
5. Integration with Spring Ecosystem: Spring Boot seamlessly integrates with the Spring ecosystem, which includes several powerful libraries and modules for building enterprise-grade applications. This allows developers to take advantage of the rich set of features provided by Spring, such as Spring Data, Spring Security, and Spring Cloud.
6. Production-Ready Features: Spring Boot includes several production-ready features, such as health checks, metrics, and monitoring, which make it easy to deploy and manage applications in a production environment.
7. Community Support: Spring Boot has a large and active community of developers who contribute to its development and provide support through forums, blogs, and other online resources.

**16. Differentiate between Spring and Spring Boot.**

Spring Boot is an extension of the Spring Framework that provides an opinionated approach to building Spring applications. Spring Boot provides a number of additional features compared to the Spring Framework, including:

1. Opinionated Configuration: Spring Boot provides a highly opinionated approach to configuration, which helps developers get started quickly without having to worry about configuring all the individual components of a Spring application.
2. Auto-Configuration: Spring Boot includes a powerful auto-configuration mechanism that automatically configures many common components of a Spring application based on the dependencies present in the classpath. This means that developers don't have to write a lot of boilerplate code to configure the application.
3. Embedded Web Servers: Spring Boot includes support for several embedded web servers, such as Tomcat, Jetty, and Undertow. This means that developers can build standalone web applications that can be run without requiring an external web server.
4. Metrics: Spring Boot provides a number of metrics and health checks out-of-the-box, which makes it easy to monitor the health and performance of an application.
5. Actuator: Spring Boot includes a powerful Actuator library that provides a number of production-ready features, such as monitoring, logging, and tracing. These features can be easily integrated into an application without requiring any additional configuration.

**17. What are the features of Spring Boot?**

Some of the key features of Spring Boot include:

1. Opinionated Configuration: Spring Boot provides a highly opinionated approach to configuration, which helps developers get started quickly without having to worry about configuring all the individual components of a Spring application.
2. Auto-Configuration: Spring Boot includes a powerful auto-configuration mechanism that automatically configures many common components of a Spring application based on the dependencies present in the classpath. This means that developers don't have to write a lot of boilerplate code to configure the application.
3. Embedded Web Servers: Spring Boot includes support for several embedded web servers, such as Tomcat, Jetty, and Undertow. This means that developers can build standalone web applications that can be run without requiring an external web server.
4. Actuator: Spring Boot includes a powerful Actuator library that provides a number of production-ready features, such as monitoring, logging, and tracing. These features can be easily integrated into an application without requiring any additional configuration.
5. Spring Data: Spring Boot provides built-in support for Spring Data, which simplifies the development of data access layers by providing a consistent programming model across different data sources.
6. Spring Security: Spring Boot includes support for Spring Security, which provides a range of security features, such as authentication, authorization, and secure communication.
7. Thymeleaf: Spring Boot includes support for Thymeleaf, a popular server-side Java template engine that enables developers to create dynamic web pages.
8. Spring Cloud: Spring Boot integrates seamlessly with Spring Cloud, a collection of tools and frameworks for building microservices-based applications.
9. Testing Frameworks: Spring Boot includes support for several testing frameworks, such as JUnit and Mockito, which make it easy to write and execute unit tests for an application.

**18.What does @SpringBootApplication annotation do internally?**

When the @SpringBootApplication annotation is used on a class, it performs the following tasks internally:

1. @Configuration: Indicates that the class is a configuration class that contains one or more Spring bean definitions. This is similar to the @Configuration annotation in the Spring Framework.
2. @EnableAutoConfiguration: Enables Spring Boot's auto-configuration feature, which automatically configures many common components of a Spring application based on the dependencies present in the classpath. This reduces the amount of boilerplate code required to configure an application.
3. @ComponentScan: Scans the specified base packages for Spring components, such as controllers, services, and repositories. This ensures that all Spring components are registered and available for use within the application.

**19. What are the effects of running Spring Boot Application as “Java Application”?**

Running a Spring Boot application as a "Java Application" is one way to launch the application from an integrated development environment (IDE) such as Eclipse or IntelliJ IDEA. When you run a Spring Boot application in this way, the IDE will launch a new Java Virtual Machine (JVM) process to run the application.

Here are some effects of running a Spring Boot application as a "Java Application":

1. Faster Development: Running the application in the IDE allows developers to make changes to the code and see the results immediately, without the need to build and deploy the application to a server.
2. Easier Debugging: Running the application in the IDE also makes it easier to debug the code by allowing developers to set breakpoints, inspect variables, and step through the code.
3. Limited Scalability: Running the application as a "Java Application" is suitable for development purposes, but it's not recommended for production environments because it is limited to running on a single JVM process.
4. Limited Resource Utilization: When running the application as a "Java Application", the application will run on the developer's machine, which means it will utilize the resources of that machine such as CPU, memory, and disk space.

**20. What is Spring Boot dependency management system?**

Spring Boot is a popular framework for building Java-based web applications. One of its key features is its dependency management system, which allows developers to easily manage the dependencies required by their applications.

Spring Boot uses a tool called Maven to manage dependencies. Maven is a build automation tool that is widely used in the Java ecosystem. It uses a file called "pom.xml" (Project Object Model) to define the dependencies required by an application.

In a Spring Boot application, the "pom.xml" file is used to declare all the required dependencies. When the application is built, Maven automatically downloads all the necessary dependencies and includes them in the application's classpath.

Spring Boot also provides a number of starter dependencies that make it easy to add common functionality to an application. For example, the "spring-boot-starter-web" dependency includes all the necessary libraries for building web applications using Spring Boot.

By using Spring Boot's dependency management system, developers can focus on writing code rather than managing dependencies. This can help streamline the development process and make it easier to build high-quality, maintainable applications.

**21. What are the possible sources of external configuration?**

External configuration is a feature of Spring Boot that allows developers to configure their applications using external properties files or environment variables. Here are the possible sources of external configuration in Spring Boot:

1. Properties Files: Spring Boot can read configuration properties from properties files in various formats, such as .properties, .yml, or .yaml. These files can be located anywhere in the classpath or in external locations, and can be named based on the application's environment.
2. Environment Variables: Spring Boot can read configuration properties from environment variables. Environment variables can be set at the system level or provided by a deployment tool such as Docker or Kubernetes.
3. Command-Line Arguments: Spring Boot can also read configuration properties from command-line arguments. These arguments can be passed when running the application from the command line or from a script.
4. Custom Property Sources: Spring Boot provides the ability to define custom property sources to read configuration properties from other sources such as a database, a remote configuration server, or a cloud-based configuration service.
5. Profiles: Spring Boot allows developers to define different configurations for different environments or scenarios by using profiles. Profiles are activated based on environment variables, command-line arguments, or the default application configuration.

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

Yes, we can change the default port of the embedded Tomcat server in Spring Boot. The default port for the embedded Tomcat server in Spring Boot is 8080, but we can change it to any other available port.

**23. Can you tell how to exclude any package without using the basePackages filter?**

Yes, we can exclude any package from component scanning in Spring Boot without using the basePackages filter. To exclude a package, we can use the "exclude" attribute of the @SpringBootApplication annotation or its aliases @SpringBootConfiguration, @EnableAutoConfiguration, and @ComponentScan.

**24. How to disable specific auto-configuration class?**

To disable a specific auto-configuration class, we can use the exclude attribute of @EnableAutoConfiguration.