|  |
| --- |
| ***What is Spring Boot ?*** |
|  |
| *Spring Boot is approach to develop Spring based application with minimal configuration.* |
| *By using Spring boot we can develop web-applications, distributed applications, standalone apps, microservice based apps, etc. with auto-config.* |
| *Spring boot is very intelligent it will identify configuration required for our application and it will provide auto-config.* |
| *Spring boot scans the class pathand find the dependancy, then it will automatically configure everyhting for us.* |

***Advantages of Spring Boot:-***

1. *Starter poms.*
2. *Dependancy version management.*
3. *Embedded Tomcat.*
4. *Actuator.*
5. *Security.*
6. *Auto-Configuration.*

***Version-wise features & Advantages?***

***IQ)What are transitive dependency in Spring Boot? 🡪All child dependencies re transitive dependencies.***

***What are Transitive Dependencies in Spring Boot?***

***🟩 Definition:***

***Transitive dependencies are the dependencies of your dependencies.  
When you include one library (dependency) in your Spring Boot project, it may automatically bring other libraries that it needs to work — those are transitive dependencies.***

***✅ Simple Example:***

***Let’s say you add this dependency in your pom.xml:***

***<dependency>***

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

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

***</dependency>***

***What happens behind the scenes?***

***This starter brings in:***

* ***spring-web***
* ***spring-webmvc***
* ***spring-boot***
* ***tomcat***
* ***spring-context***

***How can we increase performance of our Spring Boot applications?***

*Removing unused / unnecessary dependencies of our application, which we are not going to use very often then, those dependencies or starters should be removed. Any Transitive dependency which is unwanted can be removed or excluded by using <exclusions> <exclusion>tags.*

***If we want to remove or exclude any dependacny from parent dependency how can we do it?***

*We can use <exclusions <exclusion>/> to remove certain transitive dependency from parent one.*

*Eg From Spring starter Web we can remove tomcat dependency by using this tag.*

***What is Lombok in Spring Boot?***

*✅* ***Tricky/Indirect Version****:*

*"How do you reduce boilerplate code like getters/setters or constructors in your Spring Boot models?"  
"Can you make a DTO class clean without manually writing all methods?"*

***🟩 Definition***

***Lombok*** *is a compile-time annotation-based Java library used to* ***auto-generate*** *repetitive code like:*

* *Getters*
* *Setters*
* *Constructors*
* *toString()*
* *equals() / hashCode()*
* *Logger*
* *Builder pattern*

*It* ***inserts the code behind-the-scenes*** *during compilation, reducing clutter and increasing productivity.*

***🟩 Advantages of Lombok***

*✅ Less boilerplate → Clean, short classes  
✅ Easy to maintain and refactor  
✅ Improves readability  
✅ Integrates well with Spring Boot DTOs, Entities, Services  
✅ Supports* ***Builder pattern****, Logging, Constructors  
✅ Reduces human errors in repetitive methods*

***🟩 Most Used Lombok Annotations***

| ***Annotation*** | ***Purpose*** |
| --- | --- |
| *@Getter / @Setter* | *Auto generates getters and setters* |
| *@Data* | *Includes getters, setters, toString(), equals(), hashCode()* |
| *@NoArgsConstructor* | *Creates a no-arg constructor* |
| *@AllArgsConstructor* | *Creates a constructor with all fields* |
| *@Builder* | *Enables Builder pattern* |
| *@Slf4j* | *Injects a logger object* |

***Tricky-Interview Questions on LOMBOK:-***

***Q23. How do you handle boilerplate code in your Java projects?***

*✅* ***Short Answer:*** *Lombok helps eliminate repetitive code like getters, setters, constructors, toString(), and equals(). This improves readability, reduces errors, and enhances maintainability, especially in data-centric classes.*

***🟩 Q24. If you are working with a large team and there is a disagreement on using Lombok, how would you defend your decision to use it?***

*✅* ***Short Answer:*** *Lombok reduces boilerplate code, making the project more readable and maintainable. It accelerates development by auto-generating repetitive code, helping the team focus on business logic rather than boilerplate.*

***🟩 Q25. How do you ensure that Lombok-generated methods do not conflict with custom methods in your Java class?***

*✅* ***Short Answer:*** *If custom methods like toString() exist, Lombok’s annotations like @Data might override them. To avoid this, either exclude specific Lombok annotations or implement custom methods yourself without Lombok interference.*

***🟩 Q26. Have you encountered any issues when working with Lombok and frameworks like Spring Boot? If so, how did you resolve them?***

*✅* ***Short Answer:*** *One issue could be Lombok-generated methods not being recognized in some tools or IDEs. The solution is to enable annotation processing in your IDE settings and make sure the Lombok library is correctly integrated.*

***🟩 Q27. What are the alternatives to Lombok, and why would you choose one over the other?***

*✅* ***Short Answer:*** *Alternatives like* ***AutoValue*** *or* ***MapStruct*** *can be used for immutable objects or mapping, but Lombok is generally preferred for its simplicity and wide community support. I would use alternatives if stricter immutability or mapping logic is required.*

***🟩 Q28. Can you describe how Lombok works internally? What is happening at compile time when annotations like @Getter and @Setter are used?***

*✅* ***Short Answer:*** *Lombok uses annotation processing at compile time to generate code behind the scenes, which is added to the class files before they are compiled into bytecode. It eliminates the need to manually write getters, setters, and other methods.*

***🟩 Q29. How do you handle the situations where Lombok is not supported in certain environments, like when working with frameworks or tools that don’t support annotation processing?***

*✅* ***Short Answer:*** *In such cases, Lombok can be removed, and standard Java methods (getters/setters) can be written manually. Alternatively, annotation processing can be enabled in supported environments like IntelliJ or Eclipse.*

***🟩 Q30. If you need to refactor a class that is heavily using Lombok, how do you ensure that the refactor doesn't affect the generated code and keeps everything intact?***

*✅* ***Short Answer:*** *To refactor safely, avoid modifying Lombok-generated methods. Use explicit constructor annotations and handle custom logic outside Lombok’s scope. Test refactored code thoroughly to ensure generated methods behave as expected.*

***🟩 Q31. When using Lombok's @Builder, how do you handle validation or constraints that need to be applied during object creation?***

*✅* ***Short Answer:*** *Lombok's @Builder works with validation annotations like @NotNull. You can validate fields within a custom validation method or use a custom builder class to apply validation before object creation.*

***🟩 Q32. How would you manage compatibility issues between Lombok and Java versions, especially as Java evolves over time?***

*✅* ***Short Answer:*** *Stay updated with Lombok releases that support new Java versions. Regularly check for version compatibility and keep Lombok and your Java version synchronized. Use official Lombok documentation for guidance.*

***🟩 Q33. Does Lombok affect the performance of your application? If so, how do you mitigate any issues that arise?***

*✅* ***Short Answer:*** *Lombok generates code at compile time, so there is no runtime performance penalty. However, excessive use of @Builder and similar annotations in large classes could add minor overhead during object creation. Use Lombok judiciously.*

***🟩 Q34. Can you use Lombok in projects that require strict code reviews or coding standards? What challenges could arise, and how would you mitigate them?***

*✅* ***Short Answer:*** *Lombok can be used if the team agrees to allow it. The challenge is that Lombok hides code, which may cause issues during code reviews. To mitigate this, ensure that the team is familiar with Lombok and its benefits, and write clear documentation.*

***🟩 Q35. How would you replace Lombok annotations with manual code if Lombok is prohibited or not allowed in a project?***

*✅* ***Short Answer:*** *If Lombok is prohibited, manually write the necessary code such as getters, setters, constructors, toString(), equals(), and hashCode(). It will add verbosity but will achieve the same functionality without Lombok.*

***How Spring Boot internally works?***

***What is BootStrap class in SpringBoot?***

***@SpringBootApplication is combination of three annotation🡪 @SpringBootConfiguration, @EnableAutoConfiguration, @ComponentScan***

***I don’t want to use @SpringBootApplicaion then how can I run my SpringBootApplication🡪*** *Yes we can just by adding above three or atleast @EnableAutoConfiguration annotation.*

***Without writing any single property in application.properties how can we add properties or add configuration?***

*@SpringBootApplication(exclude=DriverManager.class) 🡪This will remove connection properties. Likewise we can do it for other properties as well.*

***Spring.autoconfigure.exclude****= property name 🡪 Written in application.properties file.*

***Actuator ->***

* *It includes a number of additional features that help us to monitor and manage the Spring Boot application*
* ***Spring Boot Actuator****is a sub-project of the Spring Boot Framework Actuator provides us several endpoints.*
* *If we want to expose all our endpoints in application then we need to add property,*

*management.endpoints.web.exposure.include=\**

*/****health*** *->It will provide status of our application (UP/DOWN) By default active)*

*/****info*** *->It will provide us information related to our application.*

*/****bean****-> It will provide us all information of the beans provided in our application*

*/****threaddump****->It will provide information about all the thread configured in application.*

*/****heapdump****->It will provide us information about the heap memory.*

*/****loggers****->It will provide information about application execution details.*

*/****configprops****->It will provide information about configuration properties of application.*

*/****mappings****->It will provide information about URL Mappings which are present inside our application.*

*/****caches****->It will provide information about cache.*

*All actuator mappings support “HTTP GET”.*

***How can you create custom endpoint of actuator?***

*@RestController,@Enpoint(id=”batch53”) 🡪 class level annotation*

*@ReadOperation @RequestMapping(method=RequestMethod.GET) 🡪method level annotation*

***What is the use / or where have you used Custom endpoint in your application?***

*What were the* ***endpoints of your application?***

***What are different starter dependencies you used in your project?***

*->Spring Boot Actuator, Spring Web, Spring Dev Tools.*

***Spring Profiling:-***

*Spring.profiles.active=test*

*Logger:-*

* *It is the process of storing application execution details in a file. By using logger we can identify bugs in our application.*
* *By using logger we can identify root cause of our problem*
* *Logger containe 3 things:-*

*a)Logger*

*b)Layout*

*c) Appender*

1. *Logger is the class which is used to print the log*
2. *It contains different methods.*
3. *Logger.trace()*
4. *Logger.debug()*
5. *Logger.info()*
6. *Logger.warn()*
7. *Logger.error() , etc*

*Layour:- It is used to print log pattern*

*Appender:- It is the destination where we can save our logs.*

*Console-Based Appender.:-It is used to display logs on console.*

*Note:- By default Spring uses Console-based Appender.*

*File based appender:- It is used to store logs in file.*

*Log levels:-*

*logging.file.name*

*logging.level.root*

*How do you check logs of your project?*

1. *We can manually check the server by going through location.*
2. *We used Splunk tool, so we had ID and password of the tool for checking the logs.*

*How did you create loggers in your project?*

*We used slf4j ->Simple Log Façade 4 Java.*

*Logs creaed by slf4j supports all the log tools.*

*Child of SLF4J.*

*SLF4J by default uses logback at backend. So by default Spring boot creates log by logback.xml*

*Log4j*

*Log4j2*

*Logback*

*JCL*

*Log stash*

*What is the difference between log4j and log4j2?*

*@RestController*

*@getMapping*

*@PostMapping*

*@RequestBody*

*@PathParameter*

*Can we use @Controller annotation instead of @RestController annotation? If yes then what are required changes in code?*

*🡪*

*How are versions of starter dependancies managed?*

*->*

*Status Codes*

*1XX ->Information*

*2XX ->Successful*

* *200 ->Ok*
* *201 ->Created*

*3XX ->Redirection*

*4XX -> Client Errors*

*5XX -> Server Errors*

*@Response Status , ResponseEntity*

*consumes, produces*

*@XmlRootElement*

***Spring Data JPA(Java Persistence API):-*** *(persist meaning-> to store permanently)*

1. *It is used to develop Spring boot Data JPA*
2. *It is used to develop persistence layer with ORM framework in application.*
3. *If we are using Data JPA then we need to provide hibernate properties in application.proerties file.*
4. *It consist 5 Interfaces*
5. *Repository*

* *Repository is the core interface and empty interface.*

1. *CRUDRepository*

* *CRUDRepository is used to perform CRUD baased operations.*
* *It consist 12 different methods.*
* ***Save(), saveAll()*** *🡪 Used for persisting data.*
* ***findAll(), fiindById(), findAllById(),*** *🡪 Used for get, fetching operation.*
* *Note:-****findById returns Optional*** *data of Optional Class. So need to use get().*
* ***Delete(), deleteById(), deleteAll(), deleteAll(entities), deleteAllById()*** *🡪Used for delete operation.*
* ***Save(entity)🡪****Used for update operation, also called as upsertmethod, combination of update and insert.*
* ***Count()*** *🡪 Used for count operation (Aggregate function)*
* ***existById()*** *🡪Used for checking given ID.*

1. *PagingAndSortingRepository*
2. *QueryByExampleExecutor*
3. *JPARepository*

***IQ)We already have hibernate then why do we need Data JPA?***

***🡪*** *Data JPA provides more wider functionality and specific methods to perform on data base than hibernate. As hibernate is implementation of JPA, it is a tool which uses JPA.*

***IQ)We already have CRUDRepository then why do we need JPA Repository?***

***IQ)How can you create stored procedure using JPARepository?***

***IQ)*** ***If we create an interface and extend it with JpaRepository we dont need to provide the implemented class to create the object because spring uses proxy design pattern to create object***

***IQ)What are HTTP Verbs and idempotent methods?***

***IQ)What is DTOClass and why do we need it?***

***Spring.mail.host=smtp.gmail.com***

***Spring.mail.port=587***

***Spring.mail.protocol=smtp***

***Spring.mail.username=***

***Spring.mail.password=***

***Spring.mail.properties.mail.smtp.auth=true***

***Spring.mail.properties.mail.smtp.starttls.enable=true (tls->transport layer security)***

***IQ)Why do we need to use Postman or Swagger like tools if we can***

***@Scheduled(fixedRate=3000) 🡪****Autopay ,auto debit-emi , auto mail sending to Admin consisting Inventory stock update is example of Scheduler*

***@EnableScheduling***

***--------------------------------------------------------------------------------------------------------------------------------------***

***Caching mechanism***

***API Documentation:- (Dependancy- springdoc-openapi-starter-ui -1.6.4)***

* *@Tag(name,description)*
* *@Operation(summary,description)*

*Springdoc.api-docs.enabled=true*

*Springdoc.api-doc.path=swagger-ui.html*

***HttpEntity, HttpeHeader (its methods)***

***RestTemplate vs WebClient***

***✅ What is Spring Boot?***

*🟩* ***Definition****:****Spring Boot*** *is an* ***opinionated framework*** *built on top of the Spring Framework that helps developers* ***create stand-alone, production-ready Spring applications*** *with minimal configuration.*

*It eliminates the need for writing boilerplate configuration by* ***auto-configuring*** *many components behind the scenes, and it comes with an* ***embedded server*** *(like Tomcat), so you don’t need to deploy WARs manually.*

***🟩 Why Spring Boot? (Need of It)***

* ***Traditional Spring applications*** *required heavy XML configuration and lots of setup.*
* *Managing* ***web.xml****,* ***DispatcherServlet****,* ***DataSource****, etc., was time-consuming.*
* *Every Spring module had to be manually configured and integrated (like Spring MVC + JPA + Security + AOP).*
* *Spring Boot solves this with:*
  + *✔️ Auto-Configuration*
  + *✔️ Opinionated Defaults*
  + *✔️ Embedded Web Server*
  + *✔️ Production-Ready Features*

*✅ “Spring Boot was created to simplify Spring application development and reduce developer effort.”*

***✅ Tricky / Indirect Interview Questions on Spring Boot***

*Interviewers may avoid directly saying “Spring Boot” to check your deeper understanding. Here are some tricky versions:*

| ***❓ Tricky Question*** | ***💡 What It's Actually Testing*** |
| --- | --- |
| *“How would you make a Spring app run without needing a server like Tomcat?”* | *Embedded server (Spring Boot)* |
| *“Can a Spring app start with just a main method?”* | *Spring Boot Application class* |
| *“How would you simplify Spring configuration for faster POC?”* | *Auto-Configuration* |
| *“What tool helps in rapid prototyping using Spring stack?”* | *Spring Boot* |
| *“How do you externalize configuration in modern Spring apps?”* | *application.properties or YAML* |
| *“How do you monitor a Spring-based microservice in production?”* | *Spring Boot Actuator* |

***🟩 Key Features of Spring Boot***

| ***Feature*** | ***Description*** |
| --- | --- |
| ***Auto Configuration*** | *Automatically configures Spring beans based on classpath dependencies.* |
| ***Embedded Server*** | *Comes with embedded Tomcat, Jetty, or Undertow.* |
| ***Starter Dependencies*** | *Provides pre-defined dependencies like spring-boot-starter-web.* |
| ***Spring Boot CLI*** | *Command-line tool to run and test Spring apps quickly.* |
| ***Actuator*** | *Provides production-ready endpoints to monitor and manage the app.* |
| ***No web.xml*** | *Servlet is configured automatically.* |
| ***Opinionated Defaults*** | *Offers sensible defaults to reduce setup time.* |

***🟩 Advantages of Spring Boot***

*✅* ***Developer Productivity*** *– Write less code, faster development.  
✅* ***Standalone Applications*** *– No need for external servers or WARs.  
✅* ***Auto Configuration*** *– Reduces manual setup.  
✅* ***Embedded Servers*** *– No need to deploy on Tomcat manually.  
✅* ***Actuator Support*** *– Built-in monitoring and metrics.  
✅* ***Easy Integration with Spring Cloud*** *– Perfect for microservices.  
✅* ***Spring Boot DevTools*** *– Enables hot reloading and productivity tools.  
✅* ***Starter Projects*** *– Reduce dependency management overhead.*

***🔴 Disadvantages of Spring Boot***

*⚠️* ***Hidden Configurations*** *– Auto-configuration may confuse beginners.  
⚠️* ***Memory Footprint*** *– Embedded servers can increase size and RAM usage.  
⚠️* ***Limited Fine-Grained Control*** *– Sometimes you need to override too much to customize things.  
⚠️* ***Overkill for Simple Apps*** *– Adds more than needed if project is very small.  
⚠️* ***Learning Curve with Actuator/Security*** *– Managing actuator endpoints and customizing security settings can be tricky.*

***🟩 Real-Time Use Case***

*🎯* ***Banking Web Portal*** *Using Spring Boot, you can create a* ***customer portal*** *that:*

* *Accepts new account requests (Spring Boot + REST)*
* *Stores user data (Spring Data JPA + MySQL)*
* *Sends emails for account creation (JavaMailSender)*
* *Tracks logs and performance (Actuator)*
* *Allows admin management via endpoints*

*@SpringBootApplication*

*public class BankingApp {*

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

*SpringApplication.run(BankingApp.class, args);*

*}*

*}*

*✅ One file, one main method, and it runs the entire application!*

***🟩 Summary for Interviews***

| ***🔑 Key Point*** | ***💬 Interview Line*** |
| --- | --- |
| *What* | *A framework to build Spring apps quickly with minimal config* |
| *Embedded Server* | *Runs without deploying to external server* |
| *Autoconfiguration* | *Beans and settings are auto-detected and initialized* |
| *Use case* | *Microservices, REST APIs, POCs, Enterprise apps* |
| *Popular Tools* | *spring-boot-starter, actuator, devtools* |

***✅ Q10. What is the difference between Spring Framework and Spring Boot?***

*🟩* ***Tricky version asked in interviews:***

* *“How do you simplify traditional Spring applications for faster development?”*
* *“Can Spring run without a deployment descriptor now?”*
* *“Which part of Spring helps you avoid writing boilerplate configuration?”*
* *“What’s the modern approach to setting up a Spring REST service quickly?”*

***🟩 Comparative Table: Spring Framework vs Spring Boot***

| ***🔸 Feature*** | ***🌿 Spring Framework*** | ***🚀 Spring Boot*** |
| --- | --- | --- |
| ***Definition*** | *A comprehensive dependency injection and aspect-oriented programming framework.* | *A rapid application development framework built on top of Spring.* |
| ***Configuration*** | *Heavy XML or Java-based configuration.* | *Auto-configuration with minimal setup.* |
| ***Boilerplate Code*** | *Requires a lot (e.g., web.xml, dispatcher-servlet.xml).* | *Almost zero boilerplate.* |
| ***Server Deployment*** | *Requires WAR file and external server (e.g., Tomcat).* | *Runs as standalone with* ***embedded servers*** *(Tomcat/Jetty).* |
| ***Main Class*** | *No single entry point (main() not used traditionally).* | *Has a main method using SpringApplication.run() to start app.* |
| ***Project Setup*** | *Manually manage dependencies using Maven/Gradle.* | *Uses* ***Spring Boot Starters*** *to manage dependencies easily.* |
| ***Monitoring/Management*** | *No built-in health checks or metrics.* | *Comes with* ***Spring Boot Actuator*** *for health, metrics, and monitoring.* |
| ***Learning Curve*** | *Steeper due to manual setup.* | *Easier for beginners and quicker for experts.* |
| ***Microservices Support*** | *Requires manual integration.* | *Designed to work seamlessly with* ***Spring Cloud****.* |
| ***DevTools*** | *Not included.* | *Includes hot reload, automatic restart via* ***Spring Boot DevTools****.* |
| ***Command Line Interface (CLI)*** | *Not available.* | *Comes with CLI to run Groovy-based Spring scripts.* |

***🟩 Summary in Simple Terms***

| ***Aspect*** | ***Spring*** | ***Spring Boot*** |
| --- | --- | --- |
| *Setup* | *Manual & verbose* | *Auto-configured* |
| *Server* | *External required* | *Embedded* |
| *Dev Time* | *Slower* | *Rapid* |
| *Monitoring* | *Manual* | *Built-in (Actuator)* |
| *Usage* | *Suitable for fine-grained enterprise control* | *Best for microservices, POCs, REST APIs* |

***🟩 Real-Time Use Case View***

| ***📌 Use Case*** | ***Prefer*** |
| --- | --- |
| *Legacy enterprise app with custom control* | *Spring* |
| *Modern REST API or microservice* | *Spring Boot* |
| *Lightweight app with full customization* | *Spring* |
| *Rapid prototyping, cloud deployment* | *Spring Boot* |

***✅ Interview Tip:***

*“Spring Boot* ***does not replace*** *Spring Framework. It* ***simplifies using it*** *by handling internal configurations so developers can focus on writing business logic instead of boilerplate setup.”*

*Find largest element of an array*

*Find 2nd largest element of an array*

*Find smallest element of an array*

*Find 2nd smallest element of an array*

*Find duplicate elements of an array*

*Find frequency of elements of an array*