Top 80 Spring Boot Interview Questions With examples

part – 1

🔥 1. What is Spring Boot and how does it differ from Spring Framework?

Answer:
SpringBoot simplifies Spring-based application development by providing auto-configuration, embedded servers, and production-ready features like health checks and metrics.

Example:

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

🔥 2. How does Spring Boot's auto-configuration work internally?

Answer:
SpringBoot uses `@EnableAutoConfiguration` and `spring.factories` file to load auto-configuration classes using `AutoConfigurationImportSelector`.

Example:
`spring-boot-autoconfigure` module defines:

```
org.springframework.boot.autoconfigure.EnableAutoConfiguration=\
com.example.autoconfig.MyAutoConfiguration
```

🔥 3. What is the difference between `@ComponentScan` and `@SpringBootApplication`?

Answer:
`@SpringBootApplication` is a meta-annotation that includes `@ComponentScan`, `@Configuration`, and `@EnableAutoConfiguration`

Example:

```
@SpringBootApplication(scanBasePackages = "com.example")
public class App {}
```

🔥 4. How to customize auto-configuration?

Answer:
Use `@ConditionalOnProperty`, `@ConditionalOnClass`, etc., in custom `@Configuration` classes.

Example:

```
@Configuration
@ConditionalOnProperty(name = "custom.feature.enabled", havingValue = "true")
public class CustomConfig {
    // Beans only created if property is enabled
}
```

🔥 5. What are Spring Boot Starters?

Answer:
Starters are dependency descriptors with a pre-defined set of dependencies for a particular feature.

Example:

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

🔥 6. How to override default properties in Spring Boot?

Answer:
Use `application.properties`, `application.yml`, or pass as JVM arguments.

Example:

```
server.port=8085
spring.datasource.url=jdbc:mysql://localhost:3306/mydb
```

---

commnet "spring" and get the complete PDF in your DM 📌

🔥 7. Explain Spring Boot Actuator and its use cases.

Answer:
Actuatorprovides production-ready features like metrics, health checks, and info endpoints.

Example:

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

Then access `/actuator/health`, `/actuator/metrics`.

🔥 8. How do you implement custom Actuator endpoints?

Answer:

```
@Component
@Endpoint(id = "custom")
public class CustomEndpoint {
    @ReadOperation
    public String getData() {
        return "Custom Endpoint Data";
    }
}
```

🔥 9. What is Spring Boot DevTools?

Answer:
Itprovides auto-restart, live reload, and configurations for faster development.

Example:
Just add dependency:

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

🔥 10. How do Profiles work in Spring Boot?

Answer:
Use `@Profile`, and `application-{profile}.properties` to configure environments.

Example:

```
@Profile("dev")
@Bean
public DataSource devDataSource() {
    // dev configuration
}
```

🔥 11. How do you secure a Spring Boot REST API using JWT?

Answer:
Usefilters to intercept requests and validate JWT tokens using libraries like `jjwt`.

Example:

```
public class JwtRequestFilter extends OncePerRequestFilter {
    @Override
    protected void doFilterInternal(...) {
        // Extract and validate JWT
    }
}
```

Register the filter in the Spring Security config.

🔥 12. How does Spring Boot load external configuration properties?

Answer:
Itloadsfrom `application.properties`, `application.yml`,environment variables, command-line args, and `@PropertySource`.

Example:

```
@ConfigurationProperties(prefix = "app.config")
public class AppConfig {
    private String name;
    // getter/setter
}
```

🔥 13. What is the difference between `@ConfigurationProperties` and `@Value`?

Answer:

- `@Value` is for single-value injection.
- `@ConfigurationProperties` binds an entire POJO to configuration.

Example:

```
@Value("${app.title}")
private String title;
```

vs.

```
@ConfigurationProperties(prefix = "app")
public class AppConfig {
    private String title;
}
```

🔥 14. How do you implement exception handling in Spring Boot?

Answer:
Use `@ControllerAdvice` and `@ExceptionHandler`.

Example:

```
@ControllerAdvice
public class GlobalExceptionHandler {
    @ExceptionHandler(ResourceNotFoundException.class)
    public ResponseEntity<?> handleNotFound(ResourceNotFoundException ex) {
        return ResponseEntity.status(HttpStatus.NOT_FOUND).body(ex.getMessage());
    }
}
```

🔥 15. How does Spring Boot handle asynchronous processing?

Answer:
Use `@EnableAsync` and `@Async`.

Example:

```
@EnableAsync
@SpringBootApplication
public class App {}

@Async
public CompletableFuture<String> processAsync() {
    return CompletableFuture.completedFuture("Done");
}
```

🔥 16. What is the difference between `@SpringBootTest` and `@WebMvcTest`?

Answer:

- `@SpringBootTest`: loads the full context.
- `@WebMvcTest`: loads only web layer (controllers).

Example:

```
@WebMvcTest(MyController.class)
public class MyControllerTest {}
```

🔥 17. How to customize embedded Tomcat server in Spring Boot?

Answer:
Implement `WebServerFactoryCustomizer`.

Example:

```
@Bean
public WebServerFactoryCustomizer<TomcatServletWebServerFactory> customizer() {
    return factory -> factory.setPort(9090);
}
```

---

commnet "spring" and get the complete PDF in your DM 📌

🔥 18. How do you perform database migrations in Spring Boot?

Answer:
Use **Flyway** or **Liquibase**.

Flyway Example:

```
<dependency>
  <groupId>org.flywaydb</groupId>
  <artifactId>flyway-core</artifactId>
</dependency>
```

Create SQL file: `db/migration/V1__init.sql`

🔥 19. How to enable and use caching in Spring Boot?

Answer:
Use `@EnableCaching`, `@Cacheable`, `@CachePut`, `@CacheEvict`.

Example:

```
@Cacheable("users")
public User getUserById(Long id) {
    return userRepo.findById(id);
}
```

🔥 20. How do you integrate Spring Boot with Kafka?

Answer:
Use `spring-kafka` dependency and configure `KafkaTemplate` and listener.

Example:

```
@KafkaListener(topics = "users", groupId = "group_id")
public void consume(String message) {
    System.out.println("Consumed: " + message);
}
```

🔥 21. How do you create a custom starter in Spring Boot?

Answer:

1. Create a library module.
2. Add `META-INF/spring.factories` or `spring/org.springframework.boot.autoconfigure.AutoConfiguration.imports`.
3. Provide autoconfiguration classes.

Example:

```
@Configuration
public class MyLibraryAutoConfiguration {
    @Bean
    public MyService myService() {
        return new MyService();
    }
}
```

`spring.factories`:

```
org.springframework.boot.autoconfigure.EnableAutoConfiguration=\
com.example.MyLibraryAutoConfiguration
```

🔥 22. How does Spring Boot handle circular dependencies?

Answer:
Springtries to resolve circular dependencies via setter injection or `@Lazy`. Constructor injection with circular dependencies will fail.

Example:

```
@Component
public class A {
    @Autowired @Lazy
    private B b;
}
```

🔥 23. How do you load multiple `application.properties` or YAML profiles?

Answer:

Use:

```
--spring.profiles.active=dev
```

Spring will load both `application.properties` and `application-dev.properties`.

Also: Use `@Profile("dev")` for conditional bean loading.

🔥 24. How do you implement multi-tenancy in Spring Boot?

Answer:

Use `AbstractRoutingDataSource` or Hibernate multi-tenancy features.

Example:

```
public class TenantRoutingDataSource extends AbstractRoutingDataSource {
    @Override
    protected Object determineCurrentLookupKey() {
        return TenantContext.getCurrentTenant();
    }
}
```

🔥 25. What is the role of `spring.factories`?

Answer:

IthelpsSpring Boot auto-discover and load configurations, listeners, and more.

Example content:

```
org.springframework.boot.autoconfigure.EnableAutoConfiguration=\
com.example.MyAutoConfiguration
```

🔥 26. How can you run code at application startup?

Answer:
Implement `CommandLineRunner` or `ApplicationRunner`.

Example:

```
@Bean
CommandLineRunner runner() {
    return args -> System.out.println("App started");
}
```

🔥 27. What is a reactive Spring Boot application?

Answer:
Reactiveapps use Project Reactor and WebFlux instead of the traditional Servlet-based stack.

Example:

```
@GetMapping("/stream")
public Flux<String> streamData() {
    return Flux.just("A", "B", "C");
}
```

🔥 28. How do you monitor and manage Spring Boot apps in production?

Answer:
UsingSpring Boot Actuator, Micrometer, and external tools like Prometheus, Grafana.

Example:

```
management.endpoints.web.exposure.include=health,metrics,prometheus
```

🔥 29. How do you test Spring Boot apps with Testcontainers?

Answer:
Use `@Testcontainers` and Docker containers to spin up real databases.

Example:

```
@Testcontainers
class MyIntegrationTest {
    @Container
    static PostgreSQLContainer<?> db = new PostgreSQLContainer<>("postgres:13");
}
```

🔥 30. How do you implement pagination and sorting in Spring Boot REST API?

Answer:
Use `Pageable` and `Sort` with Spring Data JPA.

Example:

```
@GetMapping("/users")
public Page<User> getUsers(Pageable pageable) {
    return userRepo.findAll(pageable);
}
```

🔥 31. How does Spring Boot handle dependency injection internally?

Answer:
Spring Boot uses the **Spring Framework's IoC container**,which builds the application context and injects dependenciesvia:

- Constructor injection (preferred)
- Setter or field injection (supported)

It uses CGLIB proxies and annotation scanning (`@ComponentScan`) to register and inject beans.

🔥 32. How do you implement rate limiting in Spring Boot?

Answer:
Uselibraries like **Bucket4j**, **Resilience4j**,or custom filters with counters.

Example using Bucket4j:

```
Bucket bucket = Bucket4j.builder()
    .addLimit(Bandwidth.classic(10, Refill.greedy(10, Duration.ofMinutes(1))))
    .build();

if (bucket.tryConsume(1)) {
    // proceed
} else {
    // reject request
}
```

🔥 33. How do you implement a custom `@Conditional` annotation?

Answer:

1. Create a class implementing `Condition`
2. Create a custom annotation using `@Conditional(MyCondition.class)`

Example:

```
public class MyCondition implements Condition {
    public boolean matches(...) {
        return System.getenv("ENV").equals("prod");
    }
}

@Target({ ElementType.TYPE, ElementType.METHOD })
@Retention(RetentionPolicy.RUNTIME)
@Conditional(MyCondition.class)
public @interface ConditionalOnProd {}
```

🔥 34. How do you expose a Spring Boot microservice over HTTPS?

Answer:

1. Add SSL cert (JKS file)
2. Configure `application.properties`

```
server.port=8443
server.ssl.enabled=true
server.ssl.key-store=classpath:keystore.jks
server.ssl.key-store-password=changeit
server.ssl.key-store-type=JKS
```

🔥 35. What are some techniques to reduce Spring Boot application startup time?

Answer:

- Use lazy initialization (`spring.main.lazy-initialization=true`)
- Remove unused starters
- Enable parallel classpath scanning
- Profile slow beans via Actuator

🔥 36. How do you implement a scheduled task with dynamic cron expression?

Answer:

Use `@Scheduled` with a dynamic expression from a bean or DB, using `SchedulingConfigurer`.

Example:

```
@Configuration
public class DynamicScheduler implements SchedulingConfigurer {
    @Override
    public void configureTasks(ScheduledTaskRegistrar registrar) {
        registrar.addTriggerTask(
            () -> runTask(),
            triggerContext -> {
                String cron = getCronFromDb();
                return new CronTrigger(cron).nextExecutionTime(triggerContext);
            }
        );
    }
}
```

🔥 37. How do you handle large file uploads efficiently in Spring Boot?

Answer:

- Use streaming (`StreamingResponseBody`)
- Increase buffer size in `application.properties`
- Use multipart config

Example:

```
@PostMapping("/upload")
public ResponseEntity<?> upload(@RequestParam MultipartFile file) {
    file.getInputStream(); // stream instead of reading into memory
    return ResponseEntity.ok("Uploaded");
}
```

🔥 38. How does Spring Boot support graceful shutdown?

Answer:
SinceSpring Boot 2.3+, enable graceful shutdown with:

```
server.shutdown=graceful
spring.lifecycle.timeout-per-shutdown-phase=30s
```

Implement `SmartLifecycle` or `DisposableBean` for cleanup logic.

🔥 39. How do you make Spring Boot REST APIs versioned?

Answer: Approaches:

- URI-based: `/api/v1/resource`
- Header-based: `Accept: application/vnd.company.v1+json`
- Param-based: `?version=1`

Example:

```
@RequestMapping(value = "/api/v1/users")
public class UserV1Controller {}

@RequestMapping(value = "/api/v2/users")
public class UserV2Controller {}
```

🔥 40. How do you publish and listen to application events?

Answer:
Use `ApplicationEventPublisher` and `@EventListener`.

Example:

```
public class MyEvent extends ApplicationEvent { ... }

@Component
public class MyListener {
    @EventListener
    public void handle(MyEvent event) {
        // handle event
    }
}
```

🔥 41. How do you customize error responses globally in Spring Boot?

Answer:
Use `@ControllerAdvice` with `@ExceptionHandler` or override `ErrorController`.

Example:

```
@ControllerAdvice
public class GlobalErrorHandler {
    @ExceptionHandler(Exception.class)
    public ResponseEntity<String> handleAll(Exception ex) {
        return ResponseEntity.status(500).body("Custom Error: " + ex.getMessage());
    }
}
```

🔥 42. How do you log request and response payloads in a Spring Boot application?

Answer:

Use `OncePerRequestFilter` or Spring AOP.

Example using Filter:

```
@Component
public class LoggingFilter extends OncePerRequestFilter {
    protected void doFilterInternal(...) {
        // Log request and response body
    }
}
```

🔥 43. How do you implement HATEOAS in Spring Boot REST API?

Answer:

Use `spring-boot-starter-hateoas`.

Example:

```
@GetMapping("/users/{id}")
public EntityModel<User> getUser(@PathVariable Long id) {
    User user = repo.findById(id);
    return EntityModel.of(user,
        linkTo(methodOn(UserController.class).getUser(id)).withSelfRel());
}
```

🔥 44. What is the role of `@EnableAutoConfiguration`, and how can you exclude configurations?

Answer:

`@EnableAutoConfiguration` enables auto-detection of configuration classes.

Use `exclude` to remove specific auto-configurations.

Example:

```
@SpringBootApplication(exclude = { DataSourceAutoConfiguration.class })
public class MyApp {}
```

🔥 45. How do you integrate Spring Boot with GraphQL?

Answer:
Use `spring-boot-starter-graphql`.

Example:

```
type Query {
  getUser(id: ID!): User
}

@QueryMapping
public User getUser(@Argument String id) {
    return userService.findById(id);
}
```

🔥 46. How do you implement distributed tracing in Spring Boot microservices?

Answer:

UseSpring Cloud Sleuth + Zipkin or OpenTelemetry.

Example:

```
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-starter-sleuth</artifactId>
</dependency>
```

Enable Zipkin:

```
spring.zipkin.base-url=http://localhost:9411
spring.sleuth.sampler.probability=1.0
```

🔥 47. What is the difference between `@RestController` and `@Controller` in Spring Boot?

Answer:

- `@RestController` = `@Controller` + `@ResponseBody`

- Used for REST APIs, always returns JSON/XML.

Example:

```
@RestController
public class ApiController {
    @GetMapping("/hello")
    public String hello() {
        return "Hello World";
    }
}
```

🔥 48. How do you inject a prototype bean into a singleton in Spring Boot?

Answer:
Use `ObjectProvider` or `Provider<T>`.

Example:

```
@Component
public class SingletonService {
    @Autowired
    private ObjectProvider<PrototypeBean> prototypeProvider;

    public void usePrototype() {
        PrototypeBean bean = prototypeProvider.getObject();
    }
}
```

🔥 49. How do you manage secrets (passwords, keys) in Spring Boot?

Answer:

- Use Spring Cloud Config Server with Vault
- Use environment variables
- Use encrypted property sources

Example (Vault):

```
spring.cloud.vault.token=<vault-token>
spring.cloud.vault.kv.application-name=myapp
```

🔥 50. How do you build reactive REST APIs with error handling in Spring WebFlux?

Answer:

Use `Mono.error()` and exception handling using `@ControllerAdvice`.

Example:

```
@GetMapping("/reactive")
public Mono<ResponseEntity<String>> get() {
    return Mono.just("hello")
        .map(val -> val.toUpperCase())
        .onErrorResume(ex -> Mono.error(new CustomException("Error")));
}
```

---

commnet "spring" and get the complete PDF in your DM 📌