J2EE interview questions and answers,

1-25: J2EE Basics

1. **What is J2EE?**
   - J2EE (Java 2 Enterprise Edition) is a platform for developing and deploying enterprise applications using Java technologies.
2. **What are the core components of J2EE?**
   - Servlets, JSP (JavaServer Pages), EJB (Enterprise JavaBeans), JNDI, JDBC, JMS, JTA, JavaMail, and Web Services.
3. **What is a Servlet?**
   - A Java class that handles HTTP requests and responses in a web application.
4. **What is a JSP?**
   - A technology that allows embedding Java code in HTML pages.
5. **Difference between Servlets and JSP?**
   - Servlets are Java classes; JSP is an HTML-based page with Java code. JSP is easier for designing pages, while Servlets are better for processing logic.
6. **What is an EJB?**
   - Enterprise JavaBeans (EJB) is a server-side component architecture for modular enterprise applications.
7. **What are the types of EJB?**
   - **Session Beans** (Stateless, Stateful, Singleton), **Entity Beans**, **Message-Driven Beans (MDBs)**.
8. **What is a Stateless vs. Stateful Session Bean?**
   - Stateless does not maintain a client state; Stateful maintains a session state.
9. **What is JDBC?**
   - Java Database Connectivity (JDBC) is an API for interacting with databases.
10. **What is a Connection Pool in JDBC?**
    - A cache of database connections for reuse, improving performance.
11. **What is JNDI?**
    - Java Naming and Directory Interface, used for looking up resources like DataSources, EJBs, etc.
12. **What is JMS?**

    - Java Message Service is used for asynchronous messaging between components.

13. **What is JTA?**
    - Java Transaction API, used for managing distributed transactions.
14. **What is the role of JPA in J2EE?**
    - Java Persistence API (JPA) is used for ORM (Object-Relational Mapping).
15. **Difference between Hibernate and JPA?**
    - Hibernate is an implementation of JPA; JPA is a specification.
16. **What is a WAR file?**
    - A Web Application Archive (WAR) contains Servlets, JSPs, and other web resources.
17. **What is an EAR file?**
    - Enterprise Archive (EAR) is used for packaging an enterprise application containing WAR and JAR files.
18. **What is an MVC pattern?**
    - Model-View-Controller (MVC) is a design pattern that separates business logic, UI, and control logic.
19. **What is a Filter in Servlets?**
    - A Java class that intercepts requests before reaching a Servlet.
20. **What is a Listener in Servlets?**
    - A component that listens for events like session creation or destruction.
21. **What is a Web.xml file?**
    - The deployment descriptor of a Java web application.
22. **What is Annotations in J2EE?**
    - Metadata added to Java classes for defining configurations without XML.
23. **What is the role of a DispatcherServlet in Spring?**
    - It is the front controller that handles all requests in a Spring MVC application.
24. **What are the scopes in JSP?**
    - **Page**, **Request**, **Session**, **Application**.
25. **What is the difference between forward() and sendRedirect()?**
    - `forward()` is server-side (internal), `sendRedirect()` is client-side (external).

26-50: Coding-Based Questions

How do you connect to a database in JDBC?

```
 Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/dbname",
"user", "password");
```

26.

How do you create a Servlet?

```
 @WebServlet("/hello")
public class HelloServlet extends HttpServlet {
   protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
IOException {
       response.getWriter().println("Hello, World!");
   }
}
```

27.
28. **How do you handle exceptions in Servlets?**
    - Using `try-catch`, `error-page` in web.xml, or `@ExceptionHandler` in Spring.

Write a simple JSP page.

```
 <%@ page language="java" %>
<html>
<body>
   <h1>Hello, <%= request.getParameter("name") %></h1>
</body>
</html>
```

29.
30. **What is a DAO pattern?**
    - A Data Access Object (DAO) encapsulates database interactions.

How to implement a Singleton Bean in Spring?

```
 @Component
public class SingletonBean { }
```

31.

Write a simple Spring Boot REST controller.

```
 @RestController
```

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

32.
33. **What is @Transactional in J2EE?**
    - Used to manage transactions automatically.

How do you configure a DataSource in Spring Boot?

```
 spring.datasource.url=jdbc:mysql://localhost:3306/db
spring.datasource.username=root
spring.datasource.password=pass
```

34.
35. **Difference between GET and POST?**
    - GET is idempotent and used for fetching; POST is used for modifying data.

51-75: Frameworks & Advanced Topics

51. **What is Hibernate?**
    - An ORM framework for interacting with databases.

What are Hibernate Annotations?

```
 @Entity
@Table(name = "users")
public class User { }
```

52.
53. **What is Lazy vs. Eager loading?**
    - Lazy loads when needed; Eager loads immediately.

How do you handle transactions in Hibernate?

```
 Session session = sessionFactory.openSession();
Transaction tx = session.beginTransaction();
tx.commit();
```

54.

55. **What is the difference between HashMap and ConcurrentHashMap?**
    - ConcurrentHashMap is thread-safe; HashMap is not.

76-100: Performance & Best Practices

76. **What are Microservices?**
    - Small, independent services communicating via APIs.
77. **What is API Gateway?**
    - A single entry point for multiple microservices.
78. **What is Spring Boot?**
    - A framework for creating stand-alone Spring applications.
79. **What is a Thread Pool?**
    - A collection of worker threads for managing concurrency.
80. **What is Dependency Injection?**
    - Injecting dependencies instead of creating objects manually.

Here are **20 coding-based scenario questions (81-100)** for J2EE, focusing on debugging, optimization, and best practices.

---

81. How to optimize Hibernate queries using caching?

Use **Hibernate Second-Level Cache** to avoid repeated database calls.

```
@Entity

@Cache(usage = CacheConcurrencyStrategy.READ_ONLY)

public class User {

   @Id @GeneratedValue

   private Long id;

   private String name;

}
```

---

82. How to use Criteria API to fetch data dynamically?

```
CriteriaBuilder cb = entityManager.getCriteriaBuilder();

CriteriaQuery<User> query = cb.createQuery(User.class);

Root<User> root = query.from(User.class);

query.select(root).where(cb.equal(root.get("name"), "John"));

List<User> users = entityManager.createQuery(query).getResultList();
```

---

83. How to handle a memory leak in a J2EE application?

- Close JDBC connections properly:

```
try (Connection con = dataSource.getConnection();

     Statement stmt = con.createStatement()) {

    ResultSet rs = stmt.executeQuery("SELECT * FROM users");

} catch (SQLException e) {

    e.printStackTrace();

}
```

- Use **WeakReferences** to prevent memory leaks:

```
WeakReference<String> weakRef = new WeakReference<>(new String("Memory Leak
Example"));
```

---

84. How to create a multi-threaded Servlet in J2EE?

Use **ExecutorService** to manage threads efficiently.

```
@WebServlet("/threadServlet")

public class MultiThreadedServlet extends HttpServlet {
    private final ExecutorService executor = Executors.newFixedThreadPool(5);


    protected void doGet(HttpServletRequest request, HttpServletResponse response) {

        executor.execute(() -> System.out.println("Processing request in thread: " +
Thread.currentThread().getName()));

    }

}
```

---

85. How to implement pagination in JPA?

```
int pageNumber = 0, pageSize = 10;

Pageable pageable = PageRequest.of(pageNumber, pageSize);

Page<User> userPage = userRepository.findAll(pageable);
```

---

86. How to create a REST API for file upload in Spring Boot?

```
@RestController

public class FileController {

    @PostMapping("/upload")

    public ResponseEntity<String> uploadFile(@RequestParam("file") MultipartFile file) {

        return ResponseEntity.ok("File uploaded: " + file.getOriginalFilename());

    }

}
```

---

87. How to implement role-based security in Spring Boot?

```
@Configuration

@EnableWebSecurity

public class SecurityConfig extends WebSecurityConfigurerAdapter {

   protected void configure(HttpSecurity http) throws Exception {

      http.authorizeRequests()

         .antMatchers("/admin").hasRole("ADMIN")

         .antMatchers("/user").hasAnyRole("USER", "ADMIN")

         .and().formLogin();

   }

}
```

---

88. How to implement optimistic locking in JPA?

```
@Entity

public class Product {

   @Id @GeneratedValue

   private Long id;


   @Version

   private int version;

}
```

---

89. How to call a stored procedure using JPA?

```
@Query(value = "CALL getUserById(:id)", nativeQuery = true)

User getUserById(@Param("id") Long id);
```

---

90. How to schedule a background job in Spring Boot?

```
@Component

public class ScheduledTask {

    @Scheduled(fixedRate = 5000)

    public void runTask() {

        System.out.println("Running scheduled task...");

    }

}
```

---

91. How to use WebSockets in Spring Boot?

```
@Configuration

@EnableWebSocket

public class WebSocketConfig implements WebSocketConfigurer {

    public void registerWebSocketHandlers(WebSocketHandlerRegistry registry) {

        registry.addHandler(new MyWebSocketHandler(), "/socket");

    }

}
```

---

92. How to write a custom exception handler in Spring Boot?

```
@ControllerAdvice

public class GlobalExceptionHandler {

    @ExceptionHandler(Exception.class)
    public ResponseEntity<String> handleException(Exception e) {
        return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body("Error: "
+ e.getMessage());
    }
}
```

---

93. How to configure a DataSource in Spring Boot?

```
spring.datasource.url=jdbc:mysql://localhost:3306/mydb
spring.datasource.username=root
spring.datasource.password=pass
```

---

94. How to handle transactions manually in Spring?

```
@Service
public class TransactionService {
    @Autowired private PlatformTransactionManager transactionManager;

    public void executeTransaction() {
        TransactionStatus status = transactionManager.getTransaction(new
DefaultTransactionDefinition());
        try {
            // Perform DB operations
            transactionManager.commit(status);
        } catch (Exception e) {
            transactionManager.rollback(status);
        }
    }
}
```

95. How to log SQL queries in Hibernate?

```
spring.jpa.show-sql=true

spring.jpa.properties.hibernate.format_sql=true
```

96. How to return a JSON response from a Spring Boot REST API?

```
@RestController

@RequestMapping("/api")

public class UserController {

    @GetMapping("/user")

    public ResponseEntity<User> getUser() {

        return ResponseEntity.ok(new User(1, "John"));

    }

}
```

97. How to integrate Spring Boot with Kafka?

```
@Component

public class KafkaProducer {

    @Autowired private KafkaTemplate<String, String> kafkaTemplate;


    public void sendMessage(String message) {
        kafkaTemplate.send("my_topic", message);
    }
}
```

---

98. How to call an external REST API in Spring Boot?

```
@Autowired
private RestTemplate restTemplate;

public String callApi() {
    return restTemplate.getForObject("https://api.example.com/data", String.class);
}
```

---

99. How to use @Async in Spring Boot for asynchronous processing?

```
@Service
public class AsyncService {
    @Async
    public void executeAsyncTask() {
        System.out.println("Executing async task...");
    }
}
```

---

100. How to implement a health check in Spring Boot?

```
@RestController
```

```
public class HealthCheckController {

    @GetMapping("/health")

    public String healthCheck() {

        return "Application is running";

    }

}
```