Ashu Jauhari

[Email address]

Spring Question Bank

Table of Contents

[Spring Core Set -1 3](#_Toc151739883)

[Q1. Spring Boot is used for developing? 13](#_Toc151739884)

[Q2. What is Spring Initializer? 13](#_Toc151739885)

[Q3. What does Spring Boot do to simplify the configuration of a Spring-based application? 13](#_Toc151739886)

[Q4. How can a Spring Boot application be packaged and distributed? 14](#_Toc151739887)

[Q5. Which Spring annotation is used to create RESTful web services using Spring MVC? 14](#_Toc151739888)

[Q6. @RestController annotation is a combination of the below two annotations 14](#_Toc151739889)

[Q8. Which Spring annotation is used to handle HTTP GET requests? 15](#_Toc151739890)

[Q9. Which Spring annotation is used to handle HTTP DELETE requests? 15](#_Toc151739891)

[Q10. Which Spring annotation is used to handle HTTP PUT requests? 15](#_Toc151739892)

[Q11. Which annotation do we use to mark the class as a Service class/component? 16](#_Toc151739893)

[Q12. Which is the default implementation class of the JpaRepository interface? 16](#_Toc151739894)

[Q13. Which is the default HTML template in Spring Boot 17](#_Toc151739895)

[Q14. Which starter dependency is used to develop web applications or Restful web services? 17](#_Toc151739896)

[Q15. What is the purpose of the Spring Boot Actuator? 17](#_Toc151739897)

[Q16. How can you specify the port on which a Spring Boot application runs? 18](#_Toc151739898)

[Q17. What is the purpose of the Spring Boot DevTools? 19](#_Toc151739899)

[Q18. What is the difference between Spring and Spring Boot? 19](#_Toc151739900)

[Q19. What is the default Logging implementation offered by Spring Boot? 19](#_Toc151739901)

[Q20. What is the @SpringBootApplication annotation used for? 20](#_Toc151739902)

[Q21. Which annotation Spring Boot provides for Integration testing? 20](#_Toc151739903)

[Q22. Which annotation is used to unit test Spring MVC Controllers? 20](#_Toc151739904)

[Q23. Which Spring Boot annotation is used to unit test the Spring Data JPA repository? 21](#_Toc151739905)

[Q24. What are the Spring annotations used for Exception handling? 21](#_Toc151739906)

[Q25.  Minimum Java version used for Spring Boot 3? 22](#_Toc151739907)

# Spring Core Set -1

**Q#1.** Which one is incorrect about the Spring Core module?

**(a) Using Spring Core module, we can develop a standalone application.**

**(b) The Spring Core module can be used with other modules like Spring MVC, Spring AOP etc.**

**(c) The Spring Core module provides Spring Containers.**

**(d) To develop a Spring MVC application, the Spring Core module is not required.**

Answer: (d) To develop a Spring MVC application, the Spring Core module is not required.

Explanation: The Spring Core module is always required to develop any type of Spring based application. It provides Spring Containers which are always required in order to work with a Spring based application.

**Q#2.**Which type of dependency can a Spring Container inject?

**(a) Primitive Type**

**(b) Collection Type**

**(c) Reference Type (User defined Type)**

**(d) All of the above**

Answer: (d) All of the above

Explanation: Spring container can inject dependencies if the variable’s data type of dependent class is any one of the above. For more details, kindly go through the [dependency injection article](https://javatechonline.com/spring-dependency-injection/#Which_type_of_dependencies_can_a_Spring_Container_Inject).

**Q#3.**Consider below four components A, B,C and D with @order annotations applied:

**@Component**

**@Order(-1)**

**public class A { }**

**@Component**

**@Order(5)**

**public class D { }**

**@Component**

**@Order(-24)**

**public class C { }**

**@Component**

**@Order(5)**

**public class B { }**

What will be the correct order of execution of components?

**(a) A B C D**

**(b) C A D B**

**(c) C A B D**

**(d) B D A C**

Answer: (c) C A B D

Explanation: Execution order will be as: First components with order value of negative number. Then components with order value of positive number.  
Then no order value components in alphabetical order of their names.

**Q#4.**How can we implement dependency injection in Spring?

**(a) By using XML configuration.**

**(b) By using annotations.**

**(c) By using XML configuration and annotations both.**

**(d) By implementing interfaces provided by the Spring Framework.**

Answer: (c) By using XML configuration and annotations both.

Explanation: The @Configuration annotation is used to indicate that a class contains Spring bean definitions. It enables the creation of beans using annotations, allowing developers to define and configure beans directly in Java code rather than XML configuration.

**Q#5.**What is the difference between @Component and @Bean annotations in Spring?

**(a) @Component is used for defining singleton beans, while @Bean is used for prototype beans.**

**(b) @Component is used for auto-detection of beans, while @Bean is used to define a bean manually.**

**(c) @Component is used for defining beans in XML configuration, while @Bean is used for Java-based configuration.**

**(d) @Component is used for defining controller beans, while @Bean is used for service beans.**

Answer: (b) @Component is used for auto-detection of beans, while @Bean is used for prototype beans

Explanation: The @Component annotation is used for auto-detection and automatic registration of beans, while the @Bean annotation is used for manual bean definition in Java-based configuration. @Component is typically used for singleton beans, while @Bean can be used for both singleton and prototype beans.

**Q#6.**How can we enable component scanning in Spring?

**(a) By adding <component-scan> in XML configuration.**

**(b) By using the @ComponentScan annotation on a configuration class.**

**(c) By enabling the 'component-scan' property in application.properties.**

**(d) By using the @Autowired annotation on a field.**

Answer: (b) By using the @ComponentScan annotation on a configuration class.

Explanation: The @ComponentScan annotation is used to enable component scanning in Spring. It is generally applied on a configuration class and specifies the base package(s) to scan for components. Component scanning allows Spring to automatically detect and register beans based on annotations such as @Component, @Service, or @Repository etc.

**Q#7.**Suppose there are two classes Vehicle and Engine associated with a ‘HAS-A’ relationship. In the context of Spring dependency injection, which one is correct?

**(a) The Engine will be a target class, while Vehicle will be a dependent class.**

**(b) The dependency injection concept can't be applied between the Vehicle and Engine classes.**

**(c) The Vehicle will be a target class, while Engine will be a dependent class.**

**(d) None of the above.**

Answer: (c) The Vehicle will be a target class, while Engine will be a dependent class.

Explanation: In the context of dependency injection, there are two concepts: target bean and dependent bean. A bean that takes support of other bean is called the target bean, whereas a bean that acts as a helper/supporting bean is called a dependent bean. For more details, kindly go through [dependent & target beans in dependency injection](https://javatechonline.com/spring-dependency-injection/#What_is_a_dependent_bean_in_Spring).

**Q#8.**Which is the correct statement about Spring Container?

**(a) It is a software program.**

**(b) It supports dependency injection and autowiring.**

**(c) It manages the bean life cycle.**

**(d) All of the above**

Answer: (d) All of the above

Explanation: The Spring container is a software program that manages the whole life cycle of a bean from its creation to destruction including dependency injection and autowiring.

**Q#9.**Which statement is incorrect about Spring Bean?

**(a) A Java class whose object is created and managed by Spring container is called a Spring Bean.**

**(b) Spring bean can be a pre-defined or user-defined or third party supplied class.**

**(c) Spring bean can be a POJO class or component class or Java bean class.**

**(d) Spring bean can be any Java class or interface.**

Answer: (d) Spring bean can be any Java class or interface. For more details on Spring Bean, kindly visit ‘[What is Spring Bean?](https://javatechonline.com/what-is-spring-bean/#What_is_Spring_Bean)‘.

Explanation: We can make any Java class as a Spring Bean, except abstract class and interface.

**Q#10.**Suppose we have a class Employee and two of its sub-classes PermanentEmployee and ContractEmployee. We want Spring container to inject PermanentEmployee in order to avoid ambiguity issue between two classes of the same type. Which code snippet can raise the ambiguity issue?

**(a) @Primary**

**@Component**

**public class PermanentEmployee extends Employee { ..... }**

**@Component**

**public class ContractEmployee extends Employee { ..... }**

**(b) <bean id="pe" class="com.dev.entity.PermanentEmployee" primary="true"/>**

**<bean id="ce" class="com.dev.entity.ContractEmployee" />**

**(c) public class XYZCompany{**

**@Autowired**

**@Primary("contractEmployee")**

**private Employee employee;**

**}**

**(d) public class XYZCompany{**

**@Autowired**

**@Qualifier("contractEmployee")**

**private Employee employee;**

**}**

Answer: (c)

Explanation: Option (c) has an incorrect use of @Primary annotation. In this situation, this can be used in sub-classes definitions.

**Q#11.**What is the purpose of the @PreDestroy annotation in Spring?

**(a) It is used to indicate a method that should be called after bean initialization.**

**(b) It is used to specify the order of bean initialization.**

**(c) It is used to indicate a method that should be called before destroying the bean.**

**(d) It is used to specify a method that should be called after constructing the properties of the class.**

Answer: (c) It is used to indicate a method that should be called before destroying the bean.

Explanation: The @PreDestroy annotation is used to mark a method that should be invoked before destroying the bean.

**Q#12.**Which one of the following is an incorrect use of the @Qualifier annotation in Spring?

**(a) public class XYZCompany{**

**private Employee employee;**

**@Autowired**

**public XYZCompany(@Qualifier("contractEmployee") Employee employee){**

**this.employee=employee;**

**}**

**}**

**(b) public class XYZCompany{**

**@Autowired**

**@Qualifier("contractEmployee")**

**private Employee employee;**

**}**

**(c) public class XYZCompany{**

**private Employee employee;**

**@Autowired**

**@Qualifier("contractEmployee")**

**public XYZCompany(Employee employee){**

**this.employee=employee;**

**}**

**}**

**(d) public class XYZCompany{**

**@Autowired**

**@Qualifier("permanentEmployee")**

**private ContractEmployee employee;**

**}**

Answer: (c)

Explanation: If we have @Autowired annotation at the constructor level, we can use @Qualifier annotation at the parameter of the constructor, but not above the constructor. You may refer[autowiring in spring](https://javatechonline.com/autowiring-in-spring/#FAQ) to get more idea on this.

**Q#13.**Spring provides multiple ways to define the scope of a bean. Which one of the following is not the correct way of defining a ‘session’ scope?

**(A) @SessionScope**

**public class Employee{**

**// some properties & methods**

**}**

**(B) <bean id="employee" class="com.test.Employee" Scope="session"/>**

**(C) @Scope("session")**

**public class Employee{**

**// some properties & methods**

**}**

**(D) <bean id="employee" class="com.test.Employee" scope="session"/>**

Answer: (b)

Explanation: Option (b) has an incorrect attribute ‘Scope’. It should be ‘scope’. All Other options are correct.

**Q#14.**How can you define a bean using @Bean annotation in Spring?

**(a) By using the @Bean annotation on a method in a @Configuration class.**

**(b) By using the @Bean annotation on a method in a @Component class.**

**(c) By using the @Autowired and @Bean annotation on a field.**

**(d) By using the @Autowired and @Bean annotation on a method.**

Answer: (a) By using the @Bean annotation on a method in a @Configuration class.

Explanation: The @Bean annotation is used to define a bean in Spring while using Java-based configuration. It is generally used on a method within a class annotated with @Configuration.

**Q#15.**How to mark a class as a Spring Bean?

**(a) By using <bean> tag in the XML configuration file.**

**(b) By using @Component annotation in a Java class.**

**(c) By using @Configuration annotation in a Java class and @Bean annotation in a method in this class.**

**(d) All of the above**

Answer: (d) All of the above

Explanation: All of the above methods are used to mark a Java class as a Spring Bean.

**Q#16.**We can use @Autowired annotation at multiple places of a class. Consider some of the places given below.

1) Field      2) Setter Method      3) Parameter field      4) Normal Method

Which of the following is the correct combination of places where we can apply @Autowired annotation?

**(a) 1 and  2**

**(b) 1 and  2 and  3**

**(c) 1 and 2 and 4**

**(d) All places**

Answer: (d) All places

Explanation: We can apply @Autowired annotation in field, setter method, parameterized constructor, and normal method. Here, the method signature of the normal method must be same as the setter method.

**Q#17.**Consider the following bean definition:

**<bean id="product" class="com.dev.test.ProductBean"/>**

Which of the below bean scope is applied in this bean?

**(A) Session**

**(B) Request**

**(C) Prototype**

**(D) Singleton**

Answer: (d) Singleton

Explanation: When the scope is not specified, the session scope is applicable by default.

**Q#18.**What is the purpose of the @Configuration annotation in Spring?

**(a) It works as a substitute of @ComponentScan.**

**(b) It marks a class as a source of Bean’s definitions.**

**(c) It enables autowiring in Spring.**

**(d) It specifies the primary bean for a given type.**

Answer: (b) It marks a class as a source of Bean’s definitions.

Explanation: Using @Configuration annotation at any Java class represents that Spring container will consider it as a source of Bean’s definitions. It also defines a configuration class for Spring.

**Q#19.**What is the purpose of the @Profile annotation in Spring?

**(a) It marks a bean as eligible for autowiring.**

**(b) It associates a bean to a particular profile.**

**(c) It enables caching for a specific bean.**

**(d) It specifies the scope of a bean.**

Answer: (b) It associates a bean to a particular profile.

Explanation: By using @Profile annotation, we can make a bean belong to a particular [profile](https://docs.spring.io/spring-boot/docs/1.2.0.M1/reference/html/boot-features-profiles.html). The @Profile annotation simply takes the names of one or multiple profiles.

**Q#20.**Which of the following annotation will you use after constructing the properties of the class in the context of the life cycle methods?

**(A) @PreDestroy**

**(B) @PostConstruct**

**(C) @BeforeDestroy**

**(D) @AfterConstruct**

Answer: (b) @PostConstruct

Explanation: PostConstructs means after constructing the properties of the class. Hence @PostConstruct is the correct answer. For complete details, kindly go through the separate detailed article on [Spring Bean Life Cycle methods](https://javatechonline.com/spring-bean-life-cycle-method-examples/).

**Q#21.**What is incorrect about the BeanFactory in Spring?

**(a) It is an interface, also known as a Spring container.**

**(b) It is used to manage the life cycle of a spring bean.**

**(c) It is used to handle HTTP requests in a Spring MVC application.**

**(d) It is responsible for creating and managing Spring beans.**

Answer: (c) It is used to handle HTTP requests in a Spring MVC application.

Explanation: The BeanFactory interface, being a basic Spring Container in Spring is responsible for creating and managing Spring beans, including managing the life cycle of a spring bean.

**Q#22.**Which is the incorrect statement about the ApplicationContext and BeanFactory in Spring?

**(a) The ApplicationContext is a sub-interface of the BeanFactory.**

**(b) The ApplicationContext provides more advanced features than the BeanFactory.**

**(c) The BeanFactory supports XML-based configuration, while ApplicationContext supports XML, Java and Annotation based configuration.**

**(d) The ApplicationContext is used for testing purposes, while BeanFactory is used in production environments.**

Answer: (d) The ApplicationContext is used for testing purposes, while BeanFactory is used in production environments.

Explanation: Since ApplicationConext provides more advanced features than the BeanFactory, it is most often used in production environments.

**Q#23.**What is the purpose of the BeanPostProcessor interface in Spring?

**(a) It is used to define a custom scope for a bean.**

**(b) It is responsible for initializing beans after instantiation.**

**(c) It is used to customize the bean creation process.**

**(d) It is responsible for managing life cycle of beans.**

Answer: (c) It is used to customize the bean creation process.

Explanation: The BeanPostProcessor interface in Spring allows customization of the bean creation process. It provides hooks for performing operations before and after bean initialization, enabling developers to modify or enhance bean instances.

**Q#24.**You are using XML driven approach to specify bean definitions. Your @Autowired or @Required annotations are not working. What entry will you add in your XML configuration file?

**(a) <bean:annotation-config/>**

**(b) <context:annotation-config/>**

**(c) <context:config-annotation/>**

**(d) <context:component-scan/>**

Answer: (b) <context:annotation-config/>

Explanation: <context: annotation-config> enables functioning of some annotations such as @Required, @Autowired, @PostConstruct, @PreDestroy, @Resource.

**Q#25.**How can you define a bean’s scope in XML configuration?

**(a) By using the "scope" attribute of the <bean> element.**

**(b) By using the @Scope annotation on a bean class.**

**(c) By using the <scope> element in XML configuration.**

**(d) By using all of the approaches mentioned above**

Answer: a) By using the “scope” attribute of the <bean> element.

Explanation: In XML configuration, the scope of a bean can be defined by using the “scope” attribute of the <bean> element. The attribute value can be set to “request” in order to specify request scope.

**Q#26.**How can you define a bean’s initialization and destruction methods of bean life cycle in XML configuration?

**(a) By using the "init-method" and "destroy-method" attributes of the <bean> element.**

**(b) By using the @PostConstruct and @PreDestroy annotations.**

**(c) By using the <init-method> and <destroy-method> elements in XML configuration.**

**(d) By using the @Bean annotation with initMethod and destroyMethod attributes.**

Answer: (a) By using the “init-method” and “destroy-method” attributes of the <bean> element.

Explanation: In XML configuration, a bean’s initialization and destruction methods can be defined using the “init-method” and “destroy-method” attributes of the <bean> element. These methods will be called after bean instantiation and before bean destruction respectively.

Spring Boot – Set 1

# Q1. Spring Boot is used for developing?

1. Web applications
2. Distributed applications (Restful web services)
3. Microservices
4. All of the above

Answer : 4

# Q2. What is Spring Initializer?

1. Web-based tool to bootstrap Spring projects and generate project skeletons with the necessary configurations and dependencies.
2. A command-line tool for generating a basic Spring Boot project structure
3. A tool for initializing a Spring-based application with a specific set of dependencies
4. A tool for creating a Spring Boot application with a specific set of features

Answer : 1. Web-based tool to bootstrap Spring projects and generate project skeletons with the necessary configurations and dependencies.

Explanation:

Spring Initializr is a web-based tool provided by the Spring team to bootstrap Spring projects and generate project skeletons with the necessary configurations and dependencies. It simplifies the process of creating a new Spring-based application by providing an intuitive user interface to customize and generate project structures.

# Q3. What does Spring Boot do to simplify the configuration of a Spring-based application?

1. Automatically configures necessary beans and dependencies
2. Uses annotations to configure beans and dependencies
3. Allows for the use of XML configuration files
4. Provides a command line interface for configuring the application

## Answer:

1. Automatically configures necessary beans and dependencies

## Explanation:

Spring Boot's auto-configuration feature automatically configures the application based on the classpath dependencies and predefined conventions. It scans the classpath for libraries and frameworks and configures the application accordingly, reducing the need for manual configuration

# Q4. How can a Spring Boot application be packaged and distributed?

1. As a JAR file
2. As a WAR file
3. As a ZIP file
4. All of the above

## Answer:. All of the above

## Explanation:

Spring Boot application can be distributed as JAR, WAR, and Zip files.

# Q5. Which Spring annotation is used to create RESTful web services using Spring MVC?

1. @RestController
2. @Controller
3. @Component
4. @Rest

## Answer:

1. @RestController

## Explanation:

When you annotate a class with [**@RestController**](https://www.javaguides.net/2023/07/spring-boot-restcontroller-annotation.html), it indicates that the class is a Spring MVC REST controller responsible for handling incoming HTTP requests and generating the appropriate HTTP responses.

# Q6. @RestController annotation is a combination of the below two annotations

1. @Component and @ResponseBody annotations
2. @Controller and @ResponseBody annotations
3. @Service and @ResponseBody annotations
4. None of the above

## Answer:

2. @Controller and @ResponseBody annotations

## Explanation:

[**@RestController**](https://www.javaguides.net/2023/07/spring-boot-restcontroller-annotation.html) combines the functionality of the [**@Controller**](https://www.javaguides.net/2023/07/spring-controller-annotation-example.html) and @ResponseBody annotations into a single annotation, making it convenient for creating RESTful APIs.

# Q8. Which Spring annotation is used to handle HTTP GET requests?

1. @GetMapping
2. @PutMapping
3. @CreateMapping
4. @PostMapping

## Answer:

1. @GetMapping

## Explanation:

The @GetMapping annotation is used to map an HTTP GET request to a specific handler method in a Spring MVC controller. It is a shortcut for specifying the @RequestMapping annotation with the RequestMethod.GET method.

# Q9. Which Spring annotation is used to handle HTTP DELETE requests?

1. @GetMapping
2. @PutMapping
3. @DeleteMapping
4. @PostMapping

## Answer:

3. @DeleteMapping

# Q10. Which Spring annotation is used to handle HTTP PUT requests?

1. @GetMapping
2. @PutMapping
3. @DeleteMapping
4. @PostMapping

2. @PutMapping

The @PutMapping annotation is used to map an HTTP PUT request to a specific handler method in a Spring MVC controller. It is a shortcut for specifying the @RequestMapping annotation with the RequestMethod.PUT method.

# Q11. Which annotation do we use to mark the class as a Service class/component?

1. @Component
2. @Service
3. @Controller
4. @Repository

.Answer: @Service

## Explanation:

The @Service annotation is used in Spring Framework to mark a class as a service component. It is typically applied to classes that perform business logic, transaction management, or other services in an application.

# Q12. Which is the default implementation class of the JpaRepository interface?

1. SimpleJpaRepository class
2. JpaRepositoryImpl class
3. CustomJpaRepository class
4. DefaultJpaRepository class

## Answer:

1. SimpleJpaRepository class

## Explanation:

The SimpleJpaRepository class is a default implementation of the JpaRepository interface in Spring Data JPA. It is provided by the Spring Data JPA framework and serves as a generic repository implementation for performing common CRUD (Create, Read, Update, Delete) operations and other database operations

# Q13. Which is the default HTML template in Spring Boot

1. JSP
2. Freemarker
3. Thymeleaf
4. Groovy

## Answer:

3. Thymeleaf

## Explanation:

The default HTML template engine in Spring Boot is Thymeleaf. Thymeleaf is a modern server-side Java template engine that is designed for both web and standalone environments.

When you create a new Spring Boot project with *spring-boot-starter-thymeleaf* dependency. Spring boot enables Thymeleaf as the default HTML template engine. Spring Boot automatically configures Thymeleaf with sensible defaults, making it straightforward to start using Thymeleaf for rendering HTML templates in your application.

# Q14. Which starter dependency is used to develop web applications or Restful web services?

1. spring-boot-starter-data-jpa
2. spring-boot-starter-web
3. spring-boot-starter-rest
4. spring-boot-starter-web-dependency

## Answer:

2. spring-boot-starter-web

## Explanation:

The *spring-boot-starter-web* dependency provides the necessary components to develop Web Applications and RESTful API in a Spring Boot project, including the embedded Tomcat server, Spring Web MVC, and other required dependencies.

# Q15. What is the purpose of the Spring Boot Actuator?

1. To provide production-ready features such as monitoring and metrics
2. To provide a way to configure beans using annotations
3. To provide a way to run Spring Boot applications as a service
4. To provide a way to access the underlying database of a Spring Boot application

## Answer:

* 1. To provide production-ready features such as monitoring and metrics

## Explanation:

The purpose of the Spring Boot Actuator is to provide production-ready features and monitoring capabilities for Spring Boot applications. It offers a set of powerful endpoints and tools that enable developers and system administrators to monitor, manage, and interact with a Spring Boot application in a production environment.

# Q16. How can you specify the port on which a Spring Boot application runs?

1. By modifying the application.properties file
2. By modifying the application.yml file
3. By using the --server.port command-line option
4. All of the above

## Answer:All of the above

## Explanation:

Spring Boot provides several ways to change the server port on which your application listens. You can choose the most convenient option based on your specific requirements and preferences.

Here are some common methods to change the server port in Spring Boot:

**application.properties or application.yml:** You can modify the application.properties or application.yml file in your Spring Boot project and set the server.port property to the desired port number.

**Command-line argument:**You can pass a command-line argument *--server.port=8081* when starting the Spring Boot application. This will override the port configured in the application.properties or application.yml file.

**System properties:**You can set a system property *server.port* with the desired port number when starting the application. For example, you can use -Dserver.port=8081 as a JVM system property.

**Environment variable:**You can set an environment variable SERVER\_PORT with the desired port number. Spring Boot will automatically pick up the environment variable and use it as the server port.

**Programmatically:**If you need more dynamic control over the server port, you can programmatically configure it in your Spring Boot application code. You can use the *ServerProperties* bean to set the port programmatically.

# Q17. What is the purpose of the Spring Boot DevTools?

1. To provide a way to run Spring Boot applications in a development environment
2. To provide hot-reloading of code changes and automatic application restart
3. To provide a way to run Spring Boot applications as a service
4. To provide a way to access the underlying database of a Spring Boot application

## Answer:

2. To provide hot-reloading of code changes and automatic application restart

## Explanation:

One of the primary features of DevTools is automatic application restart. When enabled, any changes made to the application's classpath resources (such as Java files, configuration files, templates, or static resources) trigger an automatic restart of the application. This eliminates the need for manual restarts during development, saving time and improving productivity.

DevTools includes live reload functionality that automatically refreshes the browser when changes are detected in client-side resources such as HTML, CSS, or JavaScript files. It enables you to see the changes immediately without manually refreshing the page, making frontend development more efficient.

# Q18. What is the difference between Spring and Spring Boot?

1. Spring is a Java framework, while Spring Boot is a Java library
2. Spring Boot is an opinionated version of Spring, providing a set of default configurations
3. Spring Boot is a lightweight version of Spring, while Spring is a full-featured framework
4. Spring Boot is a front-end framework, while Spring is a back-end framework

## Answer:

2. Spring Boot is an opinionated version of Spring, providing a set of default configurations

## Explanation:

Spring is a comprehensive application development framework, while Spring Boot is a framework built on top of Spring that simplifies and accelerates the development of Spring-based applications. Spring Boot provides an opinionated and streamlined approach with automatic configuration, starters, and embedded servers, while Spring offers more flexibility and control over configuration and application development.

# Q19. What is the default Logging implementation offered by Spring Boot?

1. By using Log4j
2. By using Logback
3. By using SLF4J
4. By using JUL (Java Util Logging)

## Answer:

2. By using Logback

## Explanation:

By default, Spring Boot prefers Logback for its rich features and simple configuration.

# Q20. What is the @SpringBootApplication annotation used for?

1. To enable Spring Boot auto-configuration
2. To define a Spring Boot starter class
3. To define a Spring Boot controller
4. To define a Spring Boot service

## Answer:

1. To enable Spring Boot auto-configuration

## Explanation:

By including @SpringBootApplication, the class benefits from the auto-configuration, component scanning, and other features provided by Spring Boot.

# Q21. Which annotation Spring Boot provides for Integration testing?

1. @SpringBootTest annotation
2. @WebMvcTest annotation
3. @DataJpaTest annotation
4. None of the above

## Answer:

1. @SpringBootTest annotation

## Explanation:

Spring Boot provides the @SpringBootTest annotation for integration testing.

The @SpringBootTest annotation is used to specify that a particular test class is an integration test that requires the Spring Boot application context to be created. It loads the complete application context and allows you to test your application in an environment that closely resembles the production setup.

# Q22. Which annotation is used to unit test Spring MVC Controllers?

1. @SpringBootTest annotation
2. @WebMvcTest annotation
3. @DataJpaTest annotation
4. None of the above

## Answer:

2. @WebMvcTest annotation

## Explanation:

The @WebMvcTest annotation is used to unit test Spring MVC controllers in Spring Boot.

The @WebMvcTest annotation provides a convenient way to test the controllers in isolation, without loading the entire application context. It focuses on testing the web layer of the application by auto-configuring only the relevant components, such as the controllers, and their dependencies.

# Q23. Which Spring Boot annotation is used to unit test the Spring Data JPA repository?

1. @SpringBootTest annotation
2. @WebMvcTest annotation
3. @DataJpaTest annotation
4. None of the above

## Answer:

3. @DataJpaTest annotation

## Explanation:

The @DataJpaTest annotation is used to unit test Spring Data JPA repositories in Spring Boot.

The @DataJpaTest annotation provides a convenient way to test the repository layer of your application by auto-configuring only the necessary components related to Spring Data JPA. It sets up an in-memory database and initializes the EntityManager and necessary Spring Data JPA infrastructure for testing the repositories.

# Q24. What are the Spring annotations used for Exception handling?

1. @ControllerAdvice
2. @ExceptionHandler
3. @ResponseStatus
4. All of the above

## Answer:

4. All of the above

## Explanation:

Some of the commonly used Spring annotations for exception handling are:

[**@ControllerAdvice**](https://www.javaguides.net/2018/09/spring-boot-2-exception-handling-for-rest-apis.html): The @ControllerAdvice annotation is used to define a global exception handler that applies to all controllers within the application.

[**@ExceptionHandler**](https://www.javaguides.net/2023/07/spring-boot-exceptionhandler-multiple-exceptions.html): The @ExceptionHandler annotation is used to define a method that handles a specific type of exception. It is typically used in conjunction with the @ControllerAdvice annotation.

[**@ResponseStatus**](https://www.javaguides.net/2019/08/spring-boot-responsestatus-annotation.html): The @ResponseStatus annotation is used to specify the HTTP response status code that should be sent back to the client when a particular exception occurs

# Q25.  Minimum Java version used for Spring Boot 3?

1. Java 8
2. Java 11
3. Java 17
4. Java 10

## Answer:

3. Java 17

## Explanation:

To use Spring Boot 3, you need a minimum of Java 17. It will not work with Java 8 or 11. If you are using Java lower than version 17, then you need to upgrade Java first, and then upgrade/use Spring Boot 3.

**Q26. Which of the following is the default autowiring mode?**

1. Byname
2. ByType
3. No
4. Constructor

**Answer:** C) No

## Q27. How many types of IOC containers are there?

a)5

b)4

c)3

d)2

**Answer:** D) 2

 Q28. Which is better to use, ApplicationContext or BeanFactory?

1. BeanFactory
2. ApplicationContext

Answer: B) ApplicationContext

**Q29.  Spring framework's \_\_\_\_ capability allows you to insert object dependencies indirectly.**

1. Autowiring
2. Autoreferences
3. Applicationbean
4. AutoBeans

**Answer:** A) Autowiring

**Q30. Which of the following is the default autowiring mode?**

1. Byname
2. ByType
3. No
4. Constructor

**Answer:** C) No

Q31. **. Which of the following scope, restricts the bean specification to a single instance per Spring IoC container?**

1. Singleton
2. Prototype
3. Session
4. Global-session

**Answer:** A) Singleton

**Q32. Which of the following scope allows a single bean specification to have an unlimited number of object instances?**

1. Singleton
2. Prototype
3. Session
4. Global-session

**Answer:** B) Prototype

Q33. **The scope is always \_\_\_\_ by default?**

1. Singleton
2. Prototype
3. Session
4. Global-session

**Answer:** A) Singleton

JSP MCQ Set 1

1. What is full form of JSP?

**A.**Java Service Provider

**B.**Java Service Pages

**C.**Java Server Provider

**D.**Java Server Pages

#### **Explanation**

**Correct Option :D.**

1. Which of the following is true about JSP?

1. JSP technology is used to create web application.

2. The JSP pages are not easier to maintain than Servlet.

3. JSP page consists of HTML tags and JSP tags.

**A.**1 & 2

**B.**1 & 3

**C.**2 & 3

**D.**All of the above

#### **Explanation**

**Correct Option :B.**

1. Which of the following are the life cycle method of jsp?

**A.**jspInit()

**B.**\_jspService()

**C.**jspDestroy()

**D.**All of the above

#### **Explanation**

**Correct Option :D.**

1. Request processing of JSP is done by calling which method?

**A.**jspInit()

**B.**\_jspService()

**C.**jspDestroy()

**D.**\_jspRequest()

#### **Explanation**

**Correct Option :B.**

1. Which of the following method helps in jsp page initialization?

**A.**jspInit()

**B.**\_jspService()

**C.**jspDestroy()

**D.**init()

#### **Explanation**

**Correct Option :A.**

1. Which of the following is correct about JSP?

1. JSP page is translated into Servlet.

2. JSP translator is a part of the web server which is responsible for translating the JSP page into Servlet.

**A.**only 1

**B.**only 2

**C.**Both 1 & 2

**D.**None

#### **Explanation**

**Correct Option :C.**

1. Which of the following folder in JSP project contains web.xml file?

**A.**META-INF

**B.**WEB-INF

**C.**context-root

**D.**Any of A & B

#### **Explanation**

**Correct Option :B.**

1. Arrange in correct sequence of JSP life cycle.

1. Instantiation

2. Request processing

3. Initialization

4. Classloading

5. Compilation of JSP Page

6. Destroy

7. Translation of JSP Page

**A.**7-5-4-1-3-2-6

**B.**6-5-4-1-3-2-7

**C.**1-5-4-7-3-2-6

**D.**7-5-4-3-1-2-6

#### **Explanation**

**Correct Option :A.**

1. Which one is the correct order of phases in JSP life cycle?

A. Initialization, Cleanup, Compilation, Execution

B. Initialization, Compilation, Cleanup, Execution

C. Compilation, Initialization, Execution, Cleanup

D. Cleanup, Compilation, Initialization, Execution

Answer» C. Compilation, Initialization, Execution, Cleanup

10. Which technology do we mix our business logic with the presentation logic?

A. Servlet

B. JSP

C. Both A and B

D. None of the above

Answer» A. Servlet

11. Which attribute specifies a JSP page that should process any exceptions thrown but not caught in the current page?

A. The ErrorPage Attribute

B. The IsErrorPage Attribute

C. Both A & B

D. None of the above

Answer» A. The ErrorPage Attribute

12. What is the extension of a JSP file?  
a) .java  
b) .jsp  
c) .html  
d) .php  
Answer: b) .jsp  
Explanation: JSP files have the extension .jsp. The Java code in a JSP file is compiled into a servlet by the JSP container.

13. Which of the following is true about JSP directives?  
a) They are executed at runtime  
b) They are used to define variables  
c) They are used to import classes  
d) They are enclosed in <%! %>  
Answer: c) They are used to import classes  
Explanation: JSP directives are instructions that are processed by the JSP container at translation time. They are used to import classes, define page attributes, and include files.

14. Which of the following is true about JSP scripting elements?  
a) They are enclosed in <%! %>  
b) They are executed at runtime  
c) They are used to output text to the response  
d) They are used to define variables  
Answer: b) They are executed at runtime  
Explanation: JSP scripting elements are code blocks that are executed at runtime. They are enclosed in <% %> and can be used to define variables, perform calculations, and output text to the response.

15. Which of the following is true about JSP expression language (EL)?  
a) It is used to access JavaBeans properties  
b) It is used to create custom tags  
c) It is used to define page attributes  
d) It is used to import classes  
Answer: a) It is used to access JavaBeans properties  
Explanation: JSP expression language (EL) is used to access JavaBeans properties in JSP pages. It simplifies the syntax for accessing Java objects by providing a concise way to reference JavaBean properties.

16. Which of the following is true about JSP custom tags?  
a) They are defined in a JSP file  
b) They are executed at runtime  
c) They are used to import classes  
d) They are defined in a tag library descriptor (TLD) file  
Answer: d) They are defined in a tag library descriptor (TLD) file  
Explanation: JSP custom tags are defined in a tag library descriptor (TLD) file, which contains information about the tag, such as its name, attributes, and implementation class. Custom tags are executed at runtime.

17. Which of the following is true about JSP implicit objects?  
a) They are created by the JSP container  
b) They are created by the developer  
c) They are used to define page attributes  
d) They are used to import classes  
Answer: a) They are created by the JSP container  
Explanation: JSP implicit objects are created by the JSP container and are available for use in JSP pages. They include objects such as request, response, session, and application.

18. Which of the following is true about JSP standard actions?  
a) They are executed at runtime  
b) They are used to import classes  
c) They are used to define variables  
d) They are enclosed in <%@ %>  
Answer: a) They are executed at runtime  
Explanation: JSP standard actions are executed at runtime and are used to perform common tasks such as forwarding requests, including files, and setting attributes. They are enclosed in <%@ %>.

19. Which of the following is not a JSP implicit object?  
a) request  
b) response  
c) pageContext  
d) httpRequest  
Answer: d) httpRequest  
Explanation: httpRequest is not a JSP implicit object. The JSP implicit objects are request, response, pageContext, session, application, out, config, and exception.

20. Which of the following JSP action tags is used to include the contents of another file?  
a) jsp:forward  
b) jsp:useBean  
c) jsp:include  
d) jsp:setProperty  
Answer: c) jsp:include  
Explanation: The jsp:include tag is used to include the contents of another file into the current JSP page.

21. Which of the following JSP action tags is used to set the value of a bean property?  
a) jsp:forward  
b) jsp:useBean  
c) jsp:include  
d) jsp:setProperty  
Answer: d) jsp:setProperty  
Explanation: The jsp:setProperty tag is used to set the value of a bean property.

22. Which of the following JSP action tags is used to forward the request to another resource?  
a) jsp:forward  
b) jsp:useBean  
c) jsp:include  
d) jsp:setProperty  
Answer: a) jsp:forward  
Explanation: The jsp:forward tag is used to forward the request to another resource.

23. Which of the following is true about JSP comments?  
a) They are enclosed in /\* \*/  
b) They are executed at runtime  
c) They are used to document the JSP page  
d) They are used to define variables  
Answer: c) They are used to document the JSP page  
Explanation: JSP comments are used to document the JSP page and are not executed at runtime. They are enclosed in <%– –%>.

24. Which of the following JSP action tags is used to create or find a bean?  
a) jsp:forward  
b) jsp:useBean  
c) jsp:include  
d) jsp:setProperty  
Answer: b) jsp:useBean  
Explanation: The jsp:useBean tag is used to create or find a bean.

25. Which of the following is true about JSP session object?  
a) It is created by the developer  
b) It is created by the JSP container  
c) It is used to define page attributes  
d) It is used to import classes  
Answer: b) It is created by the JSP container  
Explanation: JSP session object is created by the JSP container when a user first accesses a web application. It is used to store user-specific data across multiple requests.

26. Which of the following is true about JSP page directive?  
a) It is used to import classes  
b) It is used to define variables  
c) It is used to set page-level attributes  
d) It is executed at runtime  
Answer: c) It is used to set page-level attributes  
Explanation: JSP page directive is used to set page-level attributes such as the content type, language, session tracking, and error handling.

27. Which of the following is true about JSP application object?  
a) It is created by the developer  
b) It is created by the JSP container  
c) It is used to define page attributes  
d) It is used to import classes  
Answer: b) It is created by the JSP container  
Explanation: JSP application object is created by the JSP container when a web application is first loaded. It is used to store data that is accessible to all users of the application.

28. Which of the following is true about JSP expression language?  
a) It can be used in JSP pages and servlets  
b) It is used to execute Java code  
c) It is used to embed data into a JSP page  
d) It is executed at runtime  
Answer: c) It is used to embed data into a JSP page  
Explanation: JSP expression language is used to embed data into a JSP page without the need for scriptlets or other Java code. It is executed at translation time, not runtime.

29. Which of the following is true about JSP tag libraries?  
a) They are used to define custom tags  
b) They are defined in the web.xml file  
c) They are used to include external resources  
d) They are used to define page-level attributes  
Answer: a) They are used to define custom tags  
Explanation: JSP tag libraries are used to define custom tags that can be used in JSP pages. They are defined in a tag library descriptor file (TLD) and can be packaged as a JAR file.

30. sWhich of the following is not a JSP standard action?  
a) jsp:forward  
b) jsp:useBean  
c) jsp:if  
d) jsp:setProperty  
Answer: c) jsp:if  
Explanation: jsp:if is not a JSP standard action. However, it can be defined as a custom tag using JSP tag libraries.

### **Q31. Which one of the following is correct for directive in JSP?**

**Options**

A : <%@directive%>

B : <%!directive%>

C : <%directive%>

D : <%=directive%>

Microservices MCQ Set 1

1. Microservice could be considered as subset of \_\_\_\_\_\_\_\_\_\_

**A.**POA

**B.**SOA

**C.**Cloud

**D.**Java

**E.**HTTP

#### **Explanation**

**Correct Option :B.**

SOA is the correct answer

1. A monolithic application is \_\_\_\_\_\_\_\_\_

**A.**built as single unit

**B.**built to handle functionality and data of many application

**C.**both of the above

**D.**none of the above

#### **Explanation**

**Correct Option :A.**

built as single unit is the correct answer

1. Single point failure examples is \_\_\_\_\_\_\_\_\_\_

**A.**Monolithic

**B.**Microservice

**C.**both of the above

**D.**none of the above

#### **Explanation**

**Correct Option :A.**

Monolithic is the correct answer

1. Agile development and deployment is difficult in case of \_\_\_\_\_\_\_\_\_

**A.**Monolithic

**B.**Microservice

**C.**both of the above

**D.**none of the above

#### **Explanation**

**Correct Option :A.**

Monolithic is the correct answer

1. Complexity of developing testing and deploying distributed system and handling partial failures account to the disadvantage of \_\_\_\_\_\_\_\_\_

**A.**Monolithic

**B.**Microservices

**C.**both of the above

**D.**none of the above

#### **Explanation**

**Correct Option :B.**

Microservices is the correct answer

1. The decomposition of microservices is based on two categories, namely \_\_\_\_\_\_\_\_\_

**A.**Technology capability & Subdomain

**B.**Marketing & Domain

**C.**Business opportunity & Technology

**D.**Business capability & Subdomain

**E.**Business opportunity & Marketing

#### **Explanation**

**Correct Option :D.**

Business capability & Subdomain is the correct answer

1. Software built as microservices can by definition be broken down into multiple component services. The statement is:

**A.**true

**B.**false

**C.**no idea

**D.**none

#### **Explanation**

**Correct Option :A.**

true is the correct answer

1. Simple to develop test deploy scale represents \_\_\_\_\_\_\_\_\_\_

**A.**Monolithic

**B.**Microservice

**C.**both of the above

**D.**none of the above

#### **Explanation**

**Correct Option :A.**

Monolithic is the correct answer

Interview Question

* What is Loose Coupling?
* What is a Dependency?
* What is IOC (Inversion of Control)?
* What is Dependency Injection?
* Can you give few examples of Dependency Injection?
* What is Auto Wiring?
* What are the important roles of an IOC Container?
* What are Bean Factory and Application Context?
* Can you compare Bean Factory with Application Context?
* How do you create an application context with Spring?
* How does Spring know where to search for Components or Beans?
* What is a Component Scan?
* How do you define a component scan in XML and Java Configurations?
* How is it done with Spring Boot?
* What does @Component signify?
* What does @Autowired signify?
* What’s the difference Between @Controller, @Component, @Repository, and @Service Annotations in Spring?
* What is the default scope of a bean?
* Are Spring beans thread safe?
* What are the other scopes available?
* How is Spring’s singleton bean different from Gang of Four Singleton Pattern?
* What are the different types of dependency injections?
* What is setter injection?
* What is constructor injection?
* How do you choose between setter and constructor injections?
* What are the different options available to create Application Contexts for Spring?
* What is the difference between XML and Java Configurations for Spring?
* How do you choose between XML and Java Configurations for Spring?
* How does Spring do Autowiring?
* What are the different kinds of matching used by Spring for Autowiring?
* How do you debug problems with Spring Framework?
* How do you solve NoUniqueBeanDefinitionException?
* How do you solve NoSuchBeanDefinitionException?
* What is @Primary?
* What is @Qualifier?
* What is CDI (Contexts and Dependency Injection)?
* Does Spring Support CDI?
* Would you recommed to use CDI or Spring Annotations?
* What are the major features in different versions of Spring?
* What are new features in Spring Framework 4.0?
* What are new features in Spring Framework 5.0?
* What are important Spring Modules?
* What are important Spring Projects?
* What is the simplest way of ensuring that we are using single version of all Spring related dependencies?
* Name some of the design patterns used in Spring Framework?
* What do you think about Spring Framework?
* Why is Spring Popular?
* Can you give a big picture of the Spring Framework?

## Spring MVC

* What is Model 1 architecture?
* What is Model 2 architecture?
* What is Model 2 Front Controller architecture?
* Can you show an example controller method in Spring MVC?
* Can you explain a simple flow in Spring MVC?
* What is a ViewResolver?
* What is Model?
* What is ModelAndView?
* What is a RequestMapping?
* What is Dispatcher Servlet?
* How do you set up Dispatcher Servlet?
* What is a form backing object?
* How is validation done using Spring MVC?
* What is BindingResult?
* How do you map validation results to your view?
* What are Spring Form Tags?
* What is a Path Variable?
* What is a Model Attribute?
* What is a Session Attribute?
* What is a init binder?
* How do you set default date format with Spring?
* Why is Spring MVC so popular?

## Spring Boot

* What is Spring Boot?
* What are the important Goals of Spring Boot?
* What are the important Features of Spring Boot?
* Compare Spring Boot vs Spring?
* Compare Spring Boot vs Spring MVC?
* What is the importance of @SpringBootApplication?
* What is Auto Configuration?
* How can we find more information about Auto Configuration?
* What is an embedded server? Why is it important?
* What is the default embedded server with Spring Boot?
* What are the other embedded servers supported by Spring Boot?
* What are Starter Projects?
* Can you give examples of important starter projects?
* What is Starter Parent?
* What are the different things that are defined in Starter Parent?
* How does Spring Boot enforce common dependency management for all its Starter projects?
* What is Spring Initializr?
* What is application.properties?
* What are some of the important things that can customized in application.properties?
* How do you externalize configuration using Spring Boot?
* How can you add custom application properties using Spring Boot?
* What is @ConfigurationProperties?
* What is a profile?
* How do you define beans for a specific profile?
* How do you create application configuration for a specific profile?
* How do you have different configuration for different environments?
* What is Spring Boot Actuator?
* How do you monitor web services using Spring Boot Actuator?
* How do you find more information about your application envrionment using Spring Boot?
* What is a CommandLineRunner?

## Database Connectivity - JDBC, Spring JDBC & JPA

* What is Spring JDBC? How is different from JDBC?
* What is a JdbcTemplate?
* What is a RowMapper?
* What is JPA?
* What is Hibernate?
* How do you define an entity in JPA?
* What is an Entity Manager?
* What is a Persistence Context?
* How do you map relationships in JPA?
* What are the different types of relationships in JPA?
* How do you define One to One Mapping in JPA?
* How do you define One to Many Mapping in JPA?
* How do you define Many to Many Mapping in JPA?
* How do you define a datasource in a Spring Context?
* What is the use of persistence.xml
* How do you configure Entity Manager Factory and Transaction Manager?
* How do you define transaction management for Spring – Hibernate integration?

## Spring Data

* What is Spring Data?
* What is the need for Spring Data?
* What is Spring Data JPA?
* What is a CrudRepository?
* What is a PagingAndSortingRepository?

Spring Boot – Set 2

**1. What role do "starters" play in Spring Boot?**

a) They serve as entry points for applications.

b) They provide a set of default dependencies for specific tasks.

c) They define application properties.

d) They handle database migrations.

Answer : b) They provide a set of default dependencies for specific tasks.

## 2. Which file format cannot be used to configure properties in Spring Boot?

a) .XML

b) .properties

c) .json

d) .yml

Answer : json

## 3. Which annotation activates Spring Boot's auto-configuration?

a) @Activate

b) @SpringBootApplication

c) @AutoConfigure

d) @Configure

Answer: b

## 4. How does Spring Boot decide whether to autoconfigure a specific feature?

a) Based on classes available in the classpath

b) Depending on the JVM version

c) Based on the spring.xml configuration file

d) By analyzing the database schema

Answer: a) Based on classes available in the classpath

## What does Actuator in Spring Boot provide?

a) API documentation

b) Security features

c) Production-ready features like health checks and metrics

d) Data binding

Answer: c) Production-ready features like health checks and metrics

## To externalize Spring Boot's configuration, where can properties be specified?

a) application.properties

b) springboot.properties

c) config.properties

d) external.properties

Answer: a) application.properties

## What is the purpose of spring-boot-starter-parent?

a) It provides useful Maven defaults.

b) It contains core libraries for Spring Boot.

c) It adds logging dependencies.

d) It generates a basic project structure.

## How does Spring Boot simplify database configurations?

a) By eliminating the need for a database

b) By using conventions for data source URLs and drivers

c) By using JPA as the only option

d) By automating table creation

Answer: b) By using conventions for data source URLs and drivers

## . If you want to override Spring Boot's default configurations, which file should you modify?

a) spring.xml

b) boot-config.properties

c) application.yml or application.properties

d) boot.properties

Answer: c) application.yml or application.properties

## What is the primary purpose of @SpringBootApplication annotation?

a) Only to activate component scanning

b) Just to enable Spring Boot's autoconfiguration

c) To launch the embedded Tomcat server

d) A combination of component scanning, autoconfiguration, and Spring Boot's configuration

Answer: d) A combination of component scanning, autoconfiguration, and Spring Boot's configuration

## Which annotation is specifically used to indicate that a class provides Bean definitions?

a) @Bean

b) @Entity

c) @Configuration

d) @Component

Answer: c) @Configuration

## Which Spring annotation is used to create RESTful web services using Spring MVC?

a) @RestController

b) @Controller

c) @Component

d) @Rest

Answer: a) @RestController

## Which starter dependency is used to develop web applications or Restful web services?

a) spring-boot-starter-data-jpa

b) spring-boot-starter-web

c) spring-boot-starter-rest

d) spring-boot-starter-web-dependency

Answer: b) spring-boot-starter-web

### Interview Question on Microservices

### What are the advantages of microservices?

Here, are some significant advantages of using Microservices:

* Technology diversity, e., Microservices can mix easily with other frameworks, libraries,  and databases
* Fault isolation, e., a process failure should not bring the whole system down.
* Greater support for smaller and parallel team
* Independent deployment
* Deployment time reduce

### What is Spring Cloud?

Spring cloud is an Integration software that integrates with external systems. It allows microservices framework to build applications which perform restricted amounts of data processing.

### What are main differences between Microservices and Monolithic Architecture?

| **Microservices** | **Monolithic Architecture** |
| --- | --- |
| Service Startup is fast | Service startup takes time |
| Microservices are loosely coupled architecture. | Monolithic architecture is mostly tightly coupled. |
| Changes done in a single data model does not affect other Microservices. | Any changes in the data model affect the entire database |
| Microservices  focuses  on products, not projects | Monolithic put emphasize over the whole project |

### What are the challenges faced while using Microservices?

* Microservices always rely on each other. Therefore, they need to communicate with each other.
* As it is distributed system, it is a heavily involved model.
* If you are using Microservice architecture, you need to ready for operations overhead.
* You need skilled professionals to support heterogeneously distributed microservices.

### 9) In which cases microservice architecture best suited?

Microservice architecture is best suited for desktop, web, mobile devices, Smart TVs, Wearable, etc.

### What are the characteristics of Microservices?

* Essential messaging frameworks
* Decentralized Governance
* Easy Infrastructure automation
* Design for failure
* Infrastructure automation

### What is the meaning of OAuth?

OAuth means open authorization protocol. This protocol allows you to access the client applications on HTTP for third-party providers GitHub, Facebook, etc. It helps you to share resources stored on one site with another site without the need for their credentials.