1. Spring
2. Spring Boot
3. Spring MVC
4. JUnit
5. Mockito
6. SQL
7. Apache Maven

Spring

Q1. What Are the Benefits of Using Spring?

Spring is the most broadly used framework for the development of Java Enterprise Edition applications. The core features of Spring can be used in developing any Java application.

Q2. What Are the Benefits of Using Spring?

Spring targets to make Jakarta EE development easier. Here are the advantages of using it:

- **Lightweight:** there is a slight overhead of using the framework in development
- **Inversion of Control (IoC):** Spring container takes care of wiring dependencies of various objects, instead of creating or looking for dependent objects
- **Aspect Oriented Programming (AOP):** Spring supports AOP to separate business logic from system services
- **IoC container:** it manages Spring Bean life cycle and project specific configurations
- **MVC framework:** that is used to create web applications or RESTful web services, capable of returning XML/JSON responses
- **Transaction management:** reduces the amount of boiler-plate code in JDBC operations, file uploading, etc., either by using Java annotations or by Spring Bean XML configuration file
- **Exception Handling:**

Q3. What Spring Sub-Projects Do You Know? Describe Them Briefly.

- **Core** – a key module that provides fundamental parts of the framework, like IoC or DI
- **JDBC** – this module enables a JDBC-abstraction layer that removes the need to do JDBC coding for specific vendor databases
- **ORM integration** – provides integration layers for popular object-relational mapping APIs, such as JPA, JDO, and Hibernate
- **Web** – a web-oriented integration module, providing multipart file upload, Servlet listeners, and web-oriented application context functionalities
- **MVC framework** – a web module implementing the Model View Controller design pattern
- **AOP module** – aspect-oriented programming implementation allowing the definition of clean method-interceptors and pointcuts

Q4. What Is Dependency Injection?

Dependency Injection, an aspect of Inversion of Control (IoC), is a general concept stating that you do not create your objects manually but instead describe how they should be created. An IoC container will instantiate required classes if needed.

Q5. How to Define the Scope of a Bean?

To set Spring Bean's scope, we can use *@Scope* annotation or “scope” attribute in XML configuration files. There are five supported scopes:

- **singleton**
- **prototype**
- **request**
- **session**
- **global-session**

Q6. What Is a Spring Bean?

The Spring Beans are Java Objects that are initialized by the Spring IoC container.

Q7. How Can We Inject Beans in Spring?

A few different options exist:

- Setter Injection
- Constructor Injection
- Field Injection

Q8. Which Is the Best Way of Injecting Beans and Why?

The recommended approach is to use constructor arguments for mandatory dependencies and setters for optional ones. Constructor injection allows injecting values to immutable fields and makes testing easier.

Q9. What Is Spring Security?

Spring Security is a separate module of the Spring framework that focuses on providing authentication and authorization methods in Java applications. It also takes care of most of the common security vulnerabilities such as CSRF attacks.

Q10. How Does the Scope Prototype Work?

Scope *prototype* means that every time you call for an instance of the Bean, Spring will create a new instance and return it. This differs from the default *singleton* scope, where a single object instance is instantiated once per Spring IoC container.

Spring Boot

Q1. What Is Spring Boot?

Spring Boot is a project that provides a pre-configured set of frameworks to reduce boilerplate configuration so that you can have a Spring application up and running with the smallest amount of code.

Q2. How Can We Set up a Spring Boot Application With Maven?

We can include Spring Boot in a Maven project just like we would any other library. However, the best way is to inherit from the spring-boot-starter-parent project and declare dependencies to Spring Boot starters.

Q3. What is Spring Initializr?

Spring Initializr is a convenient way to create a Spring Boot project.

We can go to the Spring Initializr site, choose a dependency management tool (either Maven or Gradle), a language (Java, Kotlin or Groovy), a packaging scheme (Jar or War), version and dependencies, and download the project.

Q4. What Spring Boot Starters Are Available out There?

Each starter plays a role as a one-stop-shop for all the Spring technologies we need. Other required dependencies are then transitively pulled in and managed in a consistent way.

At the time of this writing, there are more than 50 starters at our disposal. The most commonly used are:

- *spring-boot-starter:* core starter, including auto-configuration support, logging, and YAML
- *spring-boot-starter-aop:* starter for aspect-oriented programming with Spring AOP and AspectJ
- *spring-boot-starter-data-jpa:* starter for using Spring Data JPA with Hibernate
- *spring-boot-starter-security:* starter for using Spring Security
- *spring-boot-starter-test:* starter for testing Spring Boot applications
- *spring-boot-starter-web:* starter for building web, including RESTful, applications using Spring MVC

Q5. How to Deploy Spring Boot Web Applications as Jar and War Files?

Traditionally, we package a web application as a WAR file, then deploy it into an external server. Doing this allows us to arrange multiple applications on the same server.

Spring tackles this problem by providing a plugin, namely spring-boot-maven-plugin, to package a web application as an executable JAR.

Q6. What Are Possible Sources of External Configuration?

We can use properties files, YAML files, environment variables, system properties, and command-line option arguments to specify configuration properties.

Q7. How to Write Integration Tests?

When running integration tests for a Spring application, we must have an *ApplicationContext*.

To make our life easier, Spring Boot provides a special annotation for testing – *@SpringBootTest*. This annotation creates an *ApplicationContext* from configuration classes indicated by its *classes* attribute.

Q8. Which Embedded Servers does Spring Boot Support, and How to Change the Default?

As of date, Spring MVC supports Tomcat, Jetty, and Undertow. Tomcat is the default application server supported by Spring Boot's web starter.

Spring WebFlux supports Reactor Netty, Tomcat, Jetty, and Undertow with Reactor Netty as default.

In Spring MVC, to change the default, let's say to Jetty, we need to exclude Tomcat and include Jetty in the dependencies.

Spring MVC

Q1. What is Spring MVC?

A Spring MVC is a Java Framework which is used to develop dynamic web applications. It implements all the basic features of a core spring framework like Inversion of Control and Dependency Injection. It follows the Model-View-Controller design pattern.

Q2. What Is the Role of the *@Autowired* Annotation?

The @Autowired annotation can be used with fields or methods for injecting a bean by type.

Q3. Explain a Model Attribute

The @ModelAttribute annotation is one of the most important annotations in Spring MVC. It binds a method parameter or a method return value to a named model attribute and then exposes it to a web view.

Q4. Explain the Difference Between *@Controller* and *@RestController*?

The main difference between the @Controller and @RestController annotations is that the @ResponseBody annotation is automatically included in the @RestController.

Q5. Describe a *PathVariable*

We can use the @PathVariable annotation as a handler method parameter in order to extract the value of a URI template variable.

Q6. Validation Using Spring MVC

Spring MVC supports JSR-303 specifications by default. We need to add JSR-303 and its implementation dependencies to our Spring MVC application. Hibernate Validator, for example, is one of the JSR-303 implementations at our disposal.

Q7. What are the *@RequestBody* and the *@ResponseBody*?

The @RequestBody annotation, used as a handler method parameter, binds the HTTP Request body to a transfer or a domain object.

When we use the @ResponseBody annotation on a handler method in the Spring MVC controller, it indicates that we'll write the return type of the method directly to the HTTP response body.

Q8. Explain *Model*, *ModelMap* and *ModelAndView*?

The Model interface defines a holder for model attributes. The ModelMap has a similar purpose, with the ability to pass a collection of values.

On the other hand, with the ModelAndView, we return the object itself. We set all the required information, like the data and the view name, in the object we're returning.

Q9. What Is *ViewResolver* in Spring?

The ViewResolver enables an application to render models in the browser – without tying the implementation to a specific view technology – by mapping view names to actual views.

Q10. What Is the Role of the *@Qualifier* Annotation?

It is used simultaneously with the @Autowired annotation to avoid confusion when multiple instances of a bean type are present.

Q11. What Is the Role of the @Required Annotation?

The @Required annotation is used on setter methods, and it indicates that the bean property that has this annotation must be populated at configuration time. Otherwise, the Spring container will throw a BeanInitializationException exception.

JUnit

Q1. What is Testing?

Testing is the process of checking the functionality of the application whether it fulfills the requirement or not.

Q2. What is Unit Testing?

Unit testing is the testing of single entity (class or method). Unit testing is very essential to every software company to give a quality product to their customers.

Q3. What is Manual testing?

Executing the test cases manually without any tool support is known as manual testing.

Q4. What is Automated testing?

Taking tool support and executing the test cases by using automation tool is known as automation testing.

Q5. What is JUnit?

JUnit is the testing framework, it is used for unit testing of Java code.

JUnit = Java + Unit Testing

Q6. What are important features of JUnit?

Import features of JUnit are:

- It is an open source framework.
- Provides Annotation to identify the test methods.
- Provides Assertions for testing expected results.
- Provides Test runners for running tests.
- JUnit tests can be run automatically and they check their own results and provide immediate feedback.
- JUnit tests can be organized into test suites containing test cases and even other test suites.
- JUnit shows test progress in a bar that is green if test is going fine and it turns red when a test fails.
- Go through the JUnit Video to get clear understanding of JUnit.

Q7. What is a Unit Test Case?

A Unit Test Case is a part of code which ensures that the another part of code (method) works as expected.

Q8. What is a test runner?

Test runner is used for executing the test cases.

Q9. What is the purpose of org.junit.Assert class?

This class provides a set of assertion methods useful for writing tests. Only failed assertions are recorded.

Q10. What is the purpose of @Test annotation in JUnit?

The Test annotation tells JUnit that the public void method to which it is attached can be run as a test case.

Q11. What happens if a JUnit Test Method is Declared as "private"?

If a JUnit test method is declared as "private", it compiles successfully. But the execution will fail. This is because JUnit requires that all test methods must be declared as "public".

Mockito

Q1. What is mockito?

Mockito is a **JAVA-based** library used for **unit testing** applications. This open-source library plays an important role in automated unit tests for the purpose of **test-driven development** or **behavior-driven development.** It uses a mock interface to add dummy functionality in the unit testing. It also uses Java reflection to create mock objects for an interface to test it.

Q2. List some benefits of Mockito?

Some of the benefits of Mockito are,

- It has support for return values.
- It supports exceptions.
- It has support for the creation of mock using annotation.
- There is no need to write mock objects on your own with Mockito.
- The test code is not broken when renaming interface method names or reordering parameters.

Q3. What is mocking in testing?

Mocking in Mockito is the way to test the functionality of the class. The mock objects do the mocking process of real service in isolation. These mock objects return the dummy data corresponding to the dummy input passed to the function. The mocking process does not require you to connect to the database or file server to test functionality.

Q4. What is difference between Assert and Verify in mockito?

Both statements are used to add validations to the test methods in the test suites but they differ in the following.

The Assert command is used to validate critical functionality. If this validation fails, then the execution of that test method is stopped and marked as failed.

In the case of **Verify** command, the test method continues the execution even after the failure of an assertion statement. The test method will be marked as failed but the execution of remaining statements of the test method is executed normally.

Q5. List some limitations of Mockito?

Some limitations of the mockito are,

- It cannot mock constructors or static methods.
- It requires Java version 6 plus to run.
- It also cannot mock equals(), hashCode() methods.
- VM mocking is only possible on VMs that are supported by Objenesis.

Download Free : Mockito Interview Questions PDF

Q6. What is use of mock() method in Mockito?

The Mock() method is used to create and inject the mocked instances. It gives you boilerplate assignments to work with these instances. The other way of creating the instances is using the **@mock** annotations.

Q7. What is ArgumentCaptor in Mockito?

ArgumentCaptor is a class that is used to capture the argument values for future assertions. This class is defined in the **org.mockito** package and can be imported from it.

Some of the methods present in this class are

- capture(),
- getValue(),
- getAllValues(), and ArgumentCaptor <U> forClass.

Q8. What is Hamcrest ?

Hamcrest is a framework used for writing customized assertion **matchers** in the Java programming language. It allows the match rules to be defined declaratively. This makes the **hamcrest** valuable in **UI validation**, **data filtering**, **writing flexible tests**, etc. It can also be used with mock objects by using adaptors. Hamcrest can also e used with **JUnit** and **TestNG**.

Q9. List some Mockito Annotations?

Some of the Mockito annotations are,

- @Mock - It is used to create and inject mocked instances.
- @Spy - It is used to create a real object and spy on the real object.
- @Captor - It is used to create an ArgumentCaptor.
- @InjectMocks - It is used to create an object of a class and insert its dependencies.

Q10. What is PowerMock?

PowerMock is a Java framework for unit testing purposes. This framework extends from other mock libraries with more powerful capabilities. It uses custom classloader and bytecode manipulation for **mocking the static methods**, **constructors**, **final classes**, **private methods**, and more. It normally lets you test the code that is regarded as untestable.

Q11. What is EasyMock?

EasyMock is a framework for creating mock objects as it uses Java reflection to create it for a given interface. It relieves the user of hand-writing mock objects as it uses a dynamic mock object generator.

Some other perks you get with EasyMock are

- exception support,
- return value support,
- refactoring scale,
- annotation support and order check support.

SQL

Q1. What is DBMS?

A Database Management System (DBMS) is a program that controls creation, maintenance and use of a database. DBMS can be termed as File Manager that manages data in a database rather than saving it in file systems.

Q2. What is RDBMS?

RDBMS stands for Relational Database Management System. RDBMS store the data into the collection of tables, which is related by common fields between the columns of the table. It also provides relational operators to manipulate the data stored into the tables.

Q3. What is SQL?

SQL stands for Structured Query Language , and it is used to communicate with the Database. This is a standard language used to perform tasks such as retrieval, updation, insertion and deletion of data from a database.

Q4. What is a Database?

Database is nothing but an organized form of data for easy access, storing, retrieval and managing of data. This is also known as structured form of data which can be accessed in many ways.

Q5. What are tables and Fields?

A table is a set of data that are organized in a model with Columns and Rows. Columns can be categorized as vertical, and Rows are horizontal. A table has specified number of column called fields but can have any number of rows which is called record.

Q6. What is a primary key?

A primary key is a combination of fields which uniquely specify a row. This is a special kind of unique key, and it has implicit NOT NULL constraint. It means, Primary key values cannot be NULL.

Q7. What is a unique key?

A Unique key constraint uniquely identified each record in the database. This provides uniqueness for the column or set of columns.

A Primary key constraint has automatic unique constraint defined on it. But not, in the case of Unique Key.

There can be many unique constraint defined per table, but only one Primary key constraint defined per table.

Q8. What is a foreign key?

A foreign key is one table which can be related to the primary key of another table. Relationship needs to be created between two tables by referencing foreign key with the primary key of another table.

Q9. What is a join?

This is a keyword used to query data from more tables based on the relationship between the fields of the tables. Keys play a major role when JOINs are used.

Q10. What are the types of join and explain each?

There are various types of join which can be used to retrieve data and it depends on the relationship between tables.

- **Inner Join.**

Inner join return rows when there is at least one match of rows between the tables.

- **Right Join.**

Right join return rows which are common between the tables and all rows of Right hand side table. Simply, it returns all the rows from the right hand side table even though there are no matches in the left hand side table.

- **Left Join.**

Left join return rows which are common between the tables and all rows of Left hand side table. Simply, it returns all the rows from Left hand side table even though there are no matches in the Right hand side table.

- **Full Join.**

Full join return rows when there are matching rows in any one of the tables. This means, it returns all the rows from the left hand side table and all the rows from the right hand side table.

Q11. What is normalization?

Normalization is the process of minimizing redundancy and dependency by organizing fields and table of a database. The main aim of Normalization is to add, delete or modify field that can be made in a single table.

Q12. What is Denormalization.

DeNormalization is a technique used to access the data from higher to lower normal forms of database. It is also process of introducing redundancy into a table by incorporating data from the related tables.

Q13. What is an Index?

An index is performance tuning method of allowing faster retrieval of records from the table. An index creates an entry for each value and it will be faster to retrieve data.

Q14. What is a relationship and what are they?

Database Relationship is defined as the connection between the tables in a database. There are various data basing relationships, and they are as follows:.

- One to One Relationship.
- One to Many Relationship.
- Many to One Relationship.
- Self-Referencing Relationship.

Q15. What is a query?

A DB query is a code written in order to get the information back from the database. Query can be designed in such a way that it matched with our expectation of the result set. Simply, a question to the Database.

Q16. What is subquery?

A subquery is a query within another query. The outer query is called as main query, and inner query is called subquery. SubQuery is always executed first, and the result of subquery is passed on to the main query.

Maven

Q1.What is Maven?

Maven is a project management and comprehension tool. Maven provides developers a complete build lifecycle framework. Development team can automate the project's build infrastructure in almost no time as Maven uses a standard directory layout and a default build lifecycle.

Q2.What is POM?

POM stands for Project Object Model. It is fundamental Unit of Work in Maven. It is an XML file. It always resides in the base directory of the project as pom.xml. It contains information about the project and various configuration details used by Maven to build the project(s).

Q3.What information does POM contain?

POM contains the some of the following configuration information −

- project dependencies
- plugins
- goals
- build profiles
- project version
- developers
- mailing list

Q4.What is Maven artifact?

An artifact is a file, usually a JAR that gets deployed to a Maven repository. A Maven build produces one or more artifacts, such as a compiled JAR and a "sources" JAR.

Each artifact has a group ID (usually a reversed domain name, like com.example.foo), an artifact ID (just a name), and a version string. The three together uniquely identify the artifact. A project's dependencies are specified as artifacts.

Q5.What is Maven Build Lifecycle?

A Build Lifecycle is a well defined sequence of phases which define the order in which the goals are to be executed. Here phase represents a stage in life cycle.

Q6.What are the phases of a Maven Build Lifecycle?

Following are the phases −

- **validate −** validate the project is correct and all necessary information is available.
- **compile −** compile the source code of the project.
- **test −** test the compiled source code using a suitable unit testing framework. These tests should not require the code be packaged or deployed
- **package −** take the compiled code and package it in its distributable format, such as a JAR.
- **integration-test −** process and deploy the package if necessary into an environment where integration tests can be run.
- **verify −** run any checks to verify the package is valid and meets quality criteria.
- **install −** install the package into the local repository, for use as a dependency in other projects locally.
- **deploy −** done in an integration or release environment, copies the final package to the remote repository for sharing with other developers and projects.

Q7.What is a Maven Repository?

A repository is a place i.e. directory where all the project jars, library jar, plugins or any other project specific artifacts are stored and can be used by Maven easily.

Q8.What types of Maven repository?

Maven repository are of three types: local, central, remote

Q9.What is SNAPSHOT in Maven?

SNAPSHOT is a special version that indicates a current development copy. Unlike regular versions, Maven checks for a new SNAPSHOT version in a remote repository for every build.

Q10.What is transitive dependency in Maven?

Transitive dependency means to avoid needing to discover and specify the libraries that your own dependencies require, and including them automatically.

Q11.What is dependency scope? Name all the dependency scope.

Dependency scope includes dependencies as per the current stage of the build. Various Dependency Scopes are −

- **compile −** This scope indicates that dependency is available in classpath of project. It is default scope.
- **provided −** This scope indicates that dependency is to be provided by JDK or web-Server/Container at runtime.
- **runtime −** This scope indicates that dependency is not required for compilation, but is required during execution.
- **test −** This scope indicates that the dependency is only available for the test compilation and execution phases.
- **system −** This scope indicates that you have to provide the system path.

- **import −** This scope is only used when dependency is of type pom. This scope indicates that the specified POM should be replaced with the dependencies in that POM's <dependencyManagement> section.