**Inversion of Control (IOC) and Dependency Injection (DI) means?**

The principle of Inversion of Control (IOC) provides the management of objects in the application and aims to minimize their dependencies. We can also consider the creation of dependencies in the project as the framework for us to do.

To give an example, the framework runs the tools necessary for a code we write to run, instead of us. The control of every part except the written code belongs to the Framework.

Advantages of the IOC:

-Allows a method to be run in isolation from its implementation.

- Allows you to easily switch between different implementations.

-Increases program modularity.

-Makes testing/writing easier as dependencies are minimized.

It can be implemented with mechanisms such as IOC, Strategy Pattern, Service Locator Pattern, Factory Pattern, **Dependency Injection (DI)**.

Dependency Injection is the most important part of Spring and is an implementation method of Inversion of Control.

Advantages of the DI:

-Instead of using the dependencies directly, it is aimed to minimize the dependency in the system by exporting these objects from outside. Thus, we are protected from changes in the dependency class.

-While it simplifies the writing of unit tests, it also increases the accuracy. One of the most important issues in software development is that the components in the software are “loosely coupled”. It is one of the important techniques that can provide this in Dependency Injection. Thus, independent classes can be tested on their own.

**Spring Bean Scopes?**

What is Bean?

Objects that form the backbone of our Spring Framework application and are managed by the Spring IOC container are called BEANs. We can consider them as reusable objects.

Some of most used Spring Bean annotations:

*@ComponentScan, @RequestBody, @Component, @PathVariable, @Repository, @RequestHeader, @Service, @Controller, @Configuration*

What is Scope?

The word Scope has meanings such as scope, area, the field of activity, and we can think that our Bean objects have a living space.

Spring framework defines 6 types of scopes: singleton, prototype, request, session, application, WebSocket

**What does @SpringBootApplication do?**

Spring Boot @SpringBootApplication annotation is used to mark a configuration class that declares one or more @Bean methods and also triggers auto-configuration and component scanning.

**What is Spring Aspect-Oriented Programming (AOP)? Where and how to use it?**

Aspect-Oriented Programming (AOP) complements Object-Oriented Programming (OOP) by providing another way of thinking about program structure. The key unit of modularity in OOP is the class, whereas in AOP the unit of modularity is the aspect. Aspects enable the modularization of concerns such as transaction management that cut across multiple types and objects.

**What is Singleton and where to use it?**

Singleton is a creational design pattern that lets you ensure that a class has only one instance, while providing a global access point to this instance. This is useful when exactly one object is needed to coordinate actions across the system. The term comes from the mathematical concept of a singleton.

**What is Spring Boot Actuator and Where to use it?**

In essence, Actuator brings production-ready features to our application. Monitoring our app, gathering metrics, understanding traffic, or the state of our database becomes trivial with this dependency. The main benefit of this library is that we can get production-grade tools without having to actually implement these features ourselves. The actuator is mainly used to expose operational information about the running application — health, metrics, info, dump, env, etc. It uses HTTP endpoints or JMX beans to enable us to interact with them. Once this dependency is on the class path, several endpoints are available for us out of the box. As with most Spring modules, we can easily configure or extend it in many ways.

**What is the primary difference between Spring and Spring Boot?**

|  |  |
| --- | --- |
| **Spring** | **Spring Boot** |
| Spring is an open-source lightweight framework widely used to develop enterprise applications. | Spring Boot is built on top of the conventional spring framework, widely used to develop REST APIs. |
| The most important feature of the Spring Framework is dependency injection. | The most important feature of the Spring Boot is Autoconfiguration. |
| It helps to create a loosely coupled application. | It helps to create a stand-alone application. |
| To run the Spring application, we need to set the server explicitly. | Spring Boot provides embedded servers such as Tomcat and Jetty etc. |
| To run the Spring application, a deployment descriptor is required. | There is no requirement for a deployment descriptor. |
| To create a Spring application, the developers write lots of code. | It reduces the lines of code. |
| It doesn’t provide support for the in-memory database. | It provides support for the in-memory database such as H2. |

**Why to use Version Control System (VCS)?**

 Version control systems are software tools that help software teams manage changes to source code over time. As development environments have accelerated, version control systems help software teams work faster and smarter.

**What are SOLID Principles? Give sample usages in Java?**

SOLID refers to five design principles in object-oriented programming, designed to reduce code rot and improve the value, function, and maintainability of software.

*Single Responsibility Principle, Open-Closed Principle, Liskov, Substitution Principle, Interface Segregation Principle, Dependency Inversion Principle*

**What is Rapid Application Development** **(RAD) model?**

The Rapid Application Development (or RAD) model is based on prototyping and iterative model with no (or less) specific planning. In general, RAD approach to software development means putting lesser emphasis on planning tasks and more emphasis on development and coming up with a prototype. In disparity to the waterfall model, which emphasizes meticulous specification and planning, the RAD approach means building on continuously evolving requirements, as more and more learnings are drawn as the development progresses.

**What is Spring Boot starter? How is it useful?**

Spring Boot provides a number of starters that allow us to add jars in the classpath. Spring Boot built-in starters make development easier and more rapid. Spring Boot Starters are the dependency descriptors.

In the Spring Boot Framework, all the starters follow a similar naming pattern: spring-boot-starter-\*, where \* denotes a particular type of application. For example, if we want to use Spring and JPA for database access, we need to include the spring-boot-starter-data-jpa dependency in our pom.xml file of the project.

**What is Caching? How can we achieve caching in Spring Boot?**

Caching is a part of temporary memory (RAM). It lies between the application and the persistence database. It stores the recently used data that reduces the number of database hits as much as possible. In other words, caching is to store data for future reference.

The primary reason for using cache is to make data access faster and less expensive. When the highly requested resource is requested multiple times, it is often beneficial for the developer to cache resources so that it can give responses quickly. Using cache in an application enhances the performance of the application. Data access from memory is always faster in comparison to fetching data from the database. It reduces both monetary cost and opportunity cost.

There are four types of caching are as follows:

In-memory Caching, Database Caching, Web server Caching, CDN Caching

@EnableCaching is a class-level annotation. We can enable caching in the Spring Boot application by using the annotation @EnableCaching. It is defined in org.springframework.cache.annotation package. It is used together with @Configuration class.

**What & How & Where & Why to logging?**

Logging: “Automatic and time-stamped documentation of events related to a particular system.”

Debugger, which is used while developing a project, cannot be used in a live project, so logging is needed. In the past, System.out, System.err, Exception.printStackTrace() were used as logging methods in Java, but this process has now become easier with the developing logging APIs.

Logging should be of a systematic nature, be controllable, and describe the status of an application.

**What is Swagger? Have you implemented it using Spring Boot?**

Swagger allows you to describe the structure of our APIs so that machines can read them. The ability of APIs to describe their own structure is the root of all awesomeness in Swagger. Well, by reading our API’s structure, it can automatically build beautiful and interactive API documentation. It can also automatically generate client libraries for our API in many languages and explore other possibilities like automated testing. Swagger does this by asking your API to return a YAML or JSON that contains a detailed description of our entire API.