HW-2

Q1 – IOC and DI means ?

Spring IoC Container is the core of Spring Framework. It creates the objects, configures and assembles their dependencies, manages their entire life cycle.

The Container uses Dependency Injection(DI) to manage the components that make up the application. It gets the information about the objects from a configuration file(XML) or Java Code or Java Annotations and Java Pojo class. These objects are called Beans. Since the Controlling of Java objects and their lifecycle is not done by the developers, hence the name **Inversion Of Control**.

Dependency Injection is a technique for passing dependencies into an object's constructor. If the object has been loaded from the container, then its dependencies will be automatically supplied by the container. This allows you to consume a dependency without having to manually create an instance. This reduces coupling and gives you greater control over the lifetime of object instances.

Q2 – Spring Bean Scopes ?

In Spring, bean scope is used to decide which type of bean instance should be returned from Spring container back to the caller. There are 5 types of bean scopes are supported :

1) Singleton : It returns a single bean instance per Spring IoC container.This single instance is stored in a cache of such singleton beans, and all subsequent requests and references for that named bean return the cached object.If no bean scope is specified in bean configuration file, default to singleton.

2) Prototype : It returns a new bean instance each time when requested. It does not store any cache version like singleton.

3) Request : It returns a single bean instance per HTTP request.

4) Session : It returns a single bean instance per HTTP session (User level session).

5) GlobalSession : It returns a single bean instance per global HTTP session. It is only valid in the context of a web-aware Spring ApplicationContext (Application level session).

Q3 – 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. It's same as declaring a class with @Configuration, @EnableAutoConfiguration and @ComponentScan annotations.

Spring Boot reduces defining of Annotation configuration. If we use @SpringBootApplication annotation at class level, then Spring Boot AutoConfigurator will automatically add all required annotations to Java Class ByteCode.

Q4 – What is Spring AOP ? Where and How to use it ?

AOP is often defined as a technique that promotes separation of concerns in a software system. Systems are composed of several components, each responsible for a specific piece of functionality. But often these components also carry additional responsibilities beyond their core functionality. System services such as logging, transaction management, and security often find their way into components whose core responsibilities is something else. These system services are commonly referred to as cross-cutting concerns because they tend to cut across multiple components in a system.

Spring AOP enables Aspect-Oriented Programming in spring applications. In AOP, aspects enable the modularization of concerns such as transaction management, logging or security that cut across multiple types and objects (often termed **crosscutting concerns**).

Q5 – What is Singleton and where to use it ?

Singleton pattern is a design pattern which restricts a class to instantiate its multiple objects. It is nothing but a way of defining a class. Class is defined in such a way that only one instance of the class is created in the complete execution of a program or project. It is used where only a single instance of a class is required to control the action throughout the execution. A singleton class shouldn't have multiple instances in any case and at any cost. Singleton classes are used for logging, driver objects, caching and thread pool, database connections.

Q6 – What is Spring Boot Actuator and Where to use it ?

Spring Boot Actuator is a sub-project of the Spring Boot Framework. It includes a number of additional features that help us to monitor and manage the Spring Boot application. It contains the actuator endpoints (the place where the resources live). We can use HTTP and JMX endpoints to manage and monitor the Spring Boot application. If we want to get production-ready features in an application, we should use the Spring Boot actuator.

There are **three** main features of Spring Boot Actuator:

Endpoint: The actuator endpoints allows us to monitor and interact with the application. Spring Boot provides a number of built-in endpoints.

Metrics: Spring Boot Actuator provides dimensional metrics by integrating with the micrometer. The micrometer is integrated into Spring Boot. It is the instrumentation library powering the delivery of application metrics from Spring. It provides vendor-neutral interfaces for timers, gauges, counters, distribution summaries**,** and long task timers with a dimensional data model.

Audit: Spring Boot provides a flexible audit framework that publishes events to an AuditEventRepository**.** It automatically publishes the authentication events if spring-security is in execution.

Q7 – 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.
To run the Spring application, we need to set the server explicitly.	Spring Boot provides embedded servers such as Tomcat and Jetty etc.
The most important feature of the Spring Framework is dependency injection.	The most important feature of the Spring Boot is Autoconfiguration.
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.

Q8 – Why to use VCS ?

With a VCS, everybody on the team is able to work absolutely freely.

The VCS will later allow you to merge all the changes into a common version.

Saving a version of your project after making changes is an essential habit.

A side-effect of using a distributed VCS like Git is that it can act as a backup; every team member has a full-blown version of the project on his disk - including the project's complete history.

Q9 – What are SOLID Principles ? Give sample usages in Java ?

SOLID design is an acronym for the following five principles:

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

The **Single Responsibility Principle** states that there should never be more than one reason for a class to change. This means that every class, or similar structure, in your code should have only one job to do.

The **Open-Closed Principle** states that classes should be open for extension but closed for modification. "Open to extension" means that you should design your classes so that new

functionality can be added as new requirements are generated. “Closed for modification” means that once you have developed a class you should never modify it, except to correct bugs.

The **Liskov Substitution Principle** applies to inheritance in such a way that the derived classes must be completely substitutable for their base classes. In other words, if class A is a subtype of class B, then we should be able to replace B with A without interrupting the behavior of the program.

The **Interface Segregation Principle** states that the larger interfaces split into smaller ones. Because the implementation classes use only the methods that are required. We should not force the client to use the methods that they do not want to use.

The **Dependency Inversion Principle** states that we must use abstraction (abstract classes and interfaces) instead of concrete implementations. High-level modules should not depend on the low-level module but both should depend on the abstraction. Because the abstraction does not depend on detail but the detail depends on abstraction.

Q10 – What is RAD model ?

RAD is a linear sequential software development process model that emphasizes a concise development cycle using an element based construction approach. If the requirements are well understood and described, and the project scope is a constraint, the RAD process enables a development team to create a fully functional system within a concise time period.

Q11 – What is Spring Boot starter ? How is it useful ?

Spring Boot simplified Spring application development. The tools it provides reduce development time considerably. No more hunting for dependencies, the starters provided by Spring simplified it by following the idea of convention rather than configuration and can help build almost any type of project specification quickly. The starters and the developer tools provide the vital strength in building Spring boot application.

Q12 – What is Caching ? How can we achive caching in Spring Boot ?

Caching is a mechanism to enhance the performance of a system. It is a temporary memory that lies between the application and the persistent database. Cache memory stores recently used data items in order to reduce the number of database hits as much as possible.

Some type of cache: In-memory caching, Database caching, Web server caching, CDN caching

Spring framework provides cache abstraction api for different cache providers. The usage of the API is very simple, yet very powerful. Today we will see the annotation based Java configuration on caching. Note that we can achieve similar functionality through XML configuration as well.

Spring Boot cache annotations: @EnableCaching, @Cacheable, @CachePut, @CacheEvict, @Caching

Q13 – What & How & Where & Why to logging ?

Logging is a powerful aid for understanding and debugging program's run-time behavior. Logs capture and persist the important data and make it available for analysis at any point in time.

It provides the complete tracing information of the application.

It records the critical failure if any occur in an application.

Q14 – What is Swagger? Have you implemented it using Spring Boot?

Swagger2 is an open source project used to generate the REST API documents for RESTful web services. It provides a user interface to access our RESTful web services via the web browser.

I implemented it by adding the necessary dependencies to the configuration file.