Week-2 Homework

1 – IOC and DI means ?

IoC (Inversion of Control) :- It's a generic term and implemented in several ways (events, delegates etc).

DI (Dependency Injection) :- DI is a sub-type of IoC and is implemented by constructor injection, setter injection or Interface injection

The advantages of IOC and DI are:

- Allows us to create loosely coupled classes

- Easy unit test writing

- Modular programs

- Manageability

2 – Spring Bean Scopes?

In Spring, objects that form the backbone of our Spring Framework application and are managed by the Spring IOC container are called Beans and our beans have a lifecycle. In order for it to perform the operations we want, we need to determine the scope of our Bean.

- **Singleton:** It is used to create single objects. (default)

- **Prototype:** It is created when the bean receives a request. A different instance is created for each request.

- **Request:** Used for web applications. An instance is created every time an HTTP request arrives.

- **Session:** Used for web applications. An instance is created every time an HTTP session occurs.

- **Global-session:** This scopes a bean definition to a global HTTP session**.**

3 – What does @SpringBootApplication do ?

The @SpringBootApplication annotation is a primary annotation that combines the @EnableAutoConfiguration, @Configuration and the @ComponentScan annotations in a Spring Boot application.

The @SpringBootApplication annotation is specify the main method. The application is starting with this annotation.

The @EnableAutoConfiguration annotation automatically runs the configurations specified in the application dependencies.

The @ComponentScan annotation scans all components defined in the project.In this way models,repositories, controllers etc. all ready for use.

@Configuration – Designates the class as a configuration class for Java configuration.

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

The Aspect Oriented Programming (AOP) approach is based on the separation of cross-cutting Concerns. In every business transaction, there are some transactions that we perform jointly. For example, does the user have authorization?, Log the incoming request, validation. These are called cross cutting concerns. We solve them with annotations like @Before, @AfterReturning, @Around using Spring's AOP dependency.

5 – What is Singleton and where to use it?

Singleton design pattern is used to get a single instance of a class. The purpose is to provide a global access point to the created object. There is no need to create some objects separately for each request. For example, db connection, logging. Creating this object separately for each client will cause performance loss. Objects created with a singleton stay on a single instance and every client uses that instance.

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

Spring Boot actuator helps us to manage and monitor our application when we deploy our application to production. It contains various endpoints by which we can see where the resource is up, it has two types of endpoints which we can use to monitor our spring boot application which is JMX and HTTP, it provides us the production-ready application by which we can monitor the health of our application we should use this feature. It also applied some more things automatically in applications like metrics, health, and auditing.

7 -What is the primary difference between Spring and Spring Boot?

The main feature of the Spring Framework is dependency Injection or Inversion of Control (IOC). Spring Boot is a module of Spring Framework. It allows us to build a stand-alone application with minimal or zero configurations

Differences between them:

> - While the most important feature of Spring is dependency injection, spring boot's is Auto-configuration.
>
> - While to create a Spring application, the developers write lots of code, spring boot reduces the lines of code.
>
> - Spring doesn't support in-memory database, but spring boot provides support in-memory database.
>
> - While to run the Spring application, we need to set the server explicitly, but spring boot provides embedded server like Tomcat and Jetty.

8 – Why to use VCS?

Version control systems provide the management, reliable storage and tracking of the source code of our application.

The advantages of VSC are:

- Multiple people can work on the same project or even the same code, and these changes can be integrated into a single code file.

- We can undo changes to the codes.

- We can create different branches for different requirements and do more than one development at the same time and then collect these improvements in a single project.

- It helps teams collaborate around the world.

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

The SOLID principles of Object Oriented Design include following five principles:

- Single Responsibility Principle (SRP)

- Open Closed Design Principle

- Liskov Substitution Principle (LSP)

- Interface Segregation Principle (ISP)

- Dependency Injection or Inversion principle

10 - What is RAD model?

RAD Model or Rapid Application Development model is a software development process based on prototyping without any specific planning. In RAD model, there is less attention paid to the planning and more priority is given to the development tasks. It targets at developing software in a short span of time

11 - What is Spring Boot starter? How is it useful?

Spring boot Starter is a set of dependency complements that we can add to our application. Before Spring Boot, developers used to spend a lot of time on dependency management. Starters is designed to make the work of developers easier. Thanks to the starters, we easily add the Spring and related technologies we need to our application.

For example,

- spring-boot-starter-security: Built to use Spring security.

- spring-boot-starter-web: Used to develop REST services using Spring boot.

- spring-boot-starter-test: For testing Spring Boot applications with libraries including JUnit Jupiter, Hamcrest.

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

A cache is a high-speed data storage layer that stores a temporary subset of data.

Caching ensures efficient reuse of previously retrieved data. Its purpose is to increase data access performance by minimizing the need to access the slow storage tier. First, to activate the cache feature in Spring Boot, the @EnableCaching annotation is added to the class where the main method is located. There are several annotations for cache operations.

- **@Cacheable("cacheName")**: It defines a cache for a method's return value.

- **@CacheEvict("cacheName"):** It is used when we want to remove stale or unused data from the cache.

- **@CachePut("cacheName"):** It is used when we want to update the cache without interfering the method execution.

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

Logging is keeping a record of all data input, processes, data output, and final results in a program. Logging is important for software developing, debugging, and running. With logging, you can leave a trail of breadcrumbs so that if something goes wrong, we can determine the cause of the problem. There are a number of situations like when we are expecting an integer, we have been given a float and we can a cloud API, the service is down for maintenance, and much more. Such problems are out of control and are hard to determine.

14 - What is Swagger? Have you implemented it using Spring Boot?

Swagger (now the "Open API Initiative") is a specification and framework for describing REST APIs using a common language that everyone can understand. There are other available frameworks that have gained some popularity, such as RAML, Summation etc. but Swagger is most popular at this point of time considering its features and acceptance among the developer community.

It offers both human readable and machine readable format of documentation. It provides both JSON and UI support. JSON can be used as machine readable format and Swagger-UI is for visual display which is easy for humans to understand by just browsing the api documentation.

Not implemented yet.