W02 HomeWork – Talha Arıç, talharic@gmail.com

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.

But, Spring supports only the following two types :

Setter Injection

Setter-based DI is realized by calling setter methods on the user's beans after invoking a no-argument constructor or no-argument static factory method to instantiate their bean.

Constructor Injection

Constructor-based DI is realized by invoking a constructor with a number of arguments, each representing a collaborator.Using this we can validate that the injected beans are not null and fail fast(fail on compile time and not on run-time), so while starting application itself we get NullPointerException: bean does not exist. Constructor injection is Best practice to inject dependencies.

2 – Spring Bean Scopes ?

When defining a <bean> you have the option of declaring a scope for that bean. For example, to force Spring to produce a new bean instance each time one is needed, you should declare the bean's scope attribute to be prototype. Similarly, if you want Spring to return the same bean instance each time one is needed, you should declare the bean's scope attribute to be singleton.

The Spring Framework supports the following five scopes, three of which are available only if you use a web-aware ApplicationContext.

singleton: This scopes the bean definition to a single instance per Spring IoC container (default).

prototype: This scopes a single bean definition to have any number of object instances.

request: This scopes a bean definition to an HTTP request. Only valid in the context of a web-aware Spring ApplicationContext.

session: This scopes a bean definition to an HTTP session. Only valid in the context of a web-aware Spring ApplicationContext.

global-session: This scopes a bean definition to a global HTTP session. Only valid in the context of a web-aware Spring ApplicationContext.

3 – What does @SpringBootApplication do ?

A single @SpringBootApplication annotation can be used to enable those three features, that is:

@EnableAutoConfiguration: enable Spring Boot’s auto-configuration mechanism

@ComponentScan: enable @Component scan on the package where the application is located

@Configuration: allow to register extra beans in the context or import additional configuration classes

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

Aspect is a programming paradigm that handles the cross cutting concerns of our applications, and the paradigm's starting point is to find solutions to concerns. The motivation for using this structure is that it complies with principles such as single responsibility and don't repeat yourself.

- Our application is more flexible and easy to manage,

- Getting rid of repetitive code pattern,

- A cleaner and understandable code,

- Separation of core logic and concerns.

Logging → Logging of requests and responses coming to our service.

Transaction Management → Performing the refund process after the error that will occur in the running code cycle from the receipt of the payment.

Performance → Calculation of running times of methods.

Validation → User e-mail permission control before the e-mail to be sent.

5 – 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.

6 – 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 become 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.

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 it.

Once this dependency is on the classpath, 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.

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

Spring: Spring Framework is a widely used Java EE framework for building applications.

Spring Boot: Spring Boot Framework is widely used to develop REST APIs.

Spring: It aims to simplify Java EE development that makes developers more productive.

Spring Boot: It aims to shorten the code length and provide the easiest way to develop Web Applications.

Spring: The primary feature of the Spring Framework is dependency injection.

Spring Boot: The primary feature of Spring Boot is Autoconfiguration. It automatically configures the classes based on the requirement.

Spring: It helps to make things simpler by allowing us to develop loosely coupled applications.

Spring Boot: It helps to create a stand-alone application with less configuration.

Spring: The developer writes a lot of code (boilerplate code) to do the minimal task.

Spring Boot: It reduces boilerplate code.

Spring: To test the Spring project, we need to set up the sever explicitly.

Spring Boot: Spring Boot offers embedded server such as Jetty and Tomcat, etc.

Spring: It does not provide support for an in-memory database.

Spring Boot: It offers several plugins for working with an embedded and in-memory database such as H2.

Spring: Developers manually define dependencies for the Spring project in pom.xml.

Spring Boot: Spring Boot comes with the concept of starter in pom.xml file that internally takes care of downloading the dependencies JARs based on Spring Boot Requirement.

8 – Why to use VCS ?

Easy Modification of the codebase, Reverting Errors, Collaboration, Backup, Version Control Best Practices

9 – 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. The SOLID principles help the user develop less coupled code. If code is tightly coupled, a group of classes are dependent on one another. This should be avoided for better maintainability and readability.

One class should have one and only one responsibility

Software components should be open for extension, but closed for modification

Derived types must be completely substitutable for their base types

Clients should not be forced to implement unnecessary methods which they will not use

Depend on abstractions, not on concretions

10 - 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.

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

Before Spring Boot was introduced, Spring Developers used to spend a lot of time on Dependency management. Spring Boot Starters were introduced to solve this problem so that the developers can spend more time on actual code than Dependencies. Spring Boot Starters are dependency descriptors that can be added under the <dependencies> section in pom.xml. There are around 50+ Spring Boot Starters for different Spring and related technologies. These starters give all the dependencies under a single name. For example, if you want to use Spring Data JPA for database access, you can include spring-boot-starter-data-jpa dependency.

Increase productivity by decreasing the Configuration time for developers.

Managing the POM is easier since the number of dependencies to be added is decreased.

Tested, Production-ready, and supported dependency configurations.

No need to remember the name and version of the dependencies.

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

The Spring Framework provides support for transparently adding caching to an application. At its core, the abstraction applies caching to methods, thus reducing the number of executions based on the information available in the cache. The caching logic is applied transparently, without any interference to the invoker. Spring Boot auto-configures the cache infrastructure as long as caching support is enabled via the @EnableCaching annotation.

In-memory Caching

In-memory caching increases the performance of the application. It is the area that is frequently used. Memcached and Redis are examples of in-memory caching. It stores key-value between application and database. Redis is an in-memory, distributed, and advanced caching tool that allows backup and restore facility. We can manage cache in distributed clusters, also.

Database Caching

Database caching is a mechanism that generates web pages on-demand (dynamically) by fetching the data from the database. It is used in a multi-tier environment that involved clients, web-application server, and database. It improves scalability and performance by distributing a query workload. The most popular database caching is the first level cache of Hibernate.

Web Server Caching

Web server caching is a mechanism that stores data for reuse. For example, a copy of a web page served by a web server. It is cached for the first time when a user visits the page. If the user requests the same next time, the cache serves a copy of the page. It avoids server form getting overloaded. Web server caching enhances the page delivery speed and reduces the work to be done by the backend server.

CDN Caching

The CDN stands for Content Delivery Network. It is a component used in modern web applications. It improves the delivery of the content by replicating commonly requested files (such as HTML Pages, stylesheet, JavaScript, images, videos, etc.) across a globally distributed set of caching servers.

It is the reason CDN becomes more popular. The CDN reduces the load on an application origin and improves the user experience. It delivers a local copy of the content from a nearby cache edge (a cache server that is closer to the end-user), or a Point of Presence (PoP).

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

When the console is closed due to the increase in in-application logging needs, due to problems such as losing all these logs.

system.err

system.out

Exception.printStackTrace()

Just like console logging methods, the necessity of having more effective methods has emerged. As a result of this, when different libraries started to appear by logging, the Jakarta team released the JCL (Jakarta Commons Logging, also known as Apache Commons Logging) project for commons logging. In this way, each logging application is adapted to a common API, so Spring Boot can use multiple libraries and use a single logger configuration for all of them.

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

Web API is rare in developing a system from those required. Because what the API does and how it should be. The high of APIs to be created for this may be too easy for how easy it is to understand the API.

Swagger supports many languages. SpringFox has been released for use with Spring Boot.

An important purpose of Swagger is that you don't have a state for RestAPI.

In another user meal that came with Swagger, our view of looking at our API information through Postaci or SoapUI in our projects has been lost. We can also perform these transactions via swagger.