# IOC and DI means ?

**Inversion of Control (IoOC)**

Inversion of Control, or IOC, is actually a design principle. In object oriented design, what we call Object Oriented Design, proposes inverting different types of control to provide a loose coupling between application classes. What is meant by control is additional responsibilities that a class has other than its main responsibility.

**Dependency Injection**

Now, Dependency Injection is a design pattern that applies the IoC principle I explained at the beginning to reverse the formation of dependent objects, that is, it is a design pattern that provides the IOC principle.

# 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.

## What is Scope?

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

## Scope Types

After defining the concept of Scope, let's take a look at Scope Types.

**Singleton:** Each Bean is a singleton by default, but is generated only once. We can think of it as in Singleton Design Pattern. Create once, use again and again :)

**Prototype:** It is created every time a request is made for the bean in question. A different instance is generated on each creation.

**Request :** Based on its name, the request bean is created when an HTTP request arrives. An active shape is covered at the HTTP request level.

**Session :** It is created when HTTP request is received in Session Scope Web Applications. An approach similar to Request Scope.

Global Session: It covers the definition of a single Bean in the lifecycle of an HTTP. Valid only in a WEB responsive Spring.

# What does @SpringBootApplication do ?

This annotation is a basic annotation that includes all three of the @Configuration, @EnbaleAutoConfiguration, @ComponentScan annotations. To summarize,

**@Configuration :** It is an annotation that performs Java-based configuration.

**@ComponentScan :** It provides automatic scanning of the components included in the project.

**@EnableAutoConfiguration :** Allows default configurations to occur automatically.

# What is Spring AOP ? Where and How to use it ?

First of all, it is one of the key components of the AOP Spring Framework. When we look at a project in general, we have non-functional codes used in different parts of our project, which are intertwined with Business-Logic.

Some of those :

- Logging

- Transaction Management

- Security

- Cache Management

- Error Management

- Performance Measurements

- Business Rules (implementation of business rules)

# What is Singleton and where to use it ?

* Singleton pattern prevents multiple instances of a class and guarantees the use of a single instance created in the virtual machine.
* The Singleton class must provide global access to reach the instance of the class.
* The constructor of the Singleton class must be private. Thus, creating a new instance from that class is prevented.
* The instance of the Singleton class must be kept as private static inside.
* It must have a public static method that returns the held instance.

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

Spring Boot Actuator automatically activates production-ready features (health check, disk usage, heap dump etc.) of applications and offers a structure that allows interacting with different HTTP endpoints.

If you want Spring Boot Actuator to be active in the project, the following Maven dependency block must be added to the pom file.

*<dependency>*

*<groupId>org.springframework.boot</groupId>*

*<artifactId>spring-boot-starter-actuator</artifactId>*

*</dependency>*

## Endpoints

/ actuator : If Spring HATEOAS is defined in your classpath, it creates a page where you can explore other endpoints,

/ auditevents : audit events in the current application

/ autoconfig : all auto-configuration definitions

/ beans : All objects managed by Spring

/configprops : All classes with @ConfigurationProperties annotation

/ dump : allows to get thread dump

/ env : Lists Spring's ConfigurableEnvironment values

/ flyway : shows any applied Flyway database migration

/ health : gives general information about the status of the application (disk or database access status, etc.)

/ info : shows the defined application information

/ loggers : log levels of defined classes

/liquibase : shows any applied Liquibase database migration

/ metrics : show metrics data for current app

/ mappings : Shows @RequestMapping definitions

/ shutdown : allows you to shut down the application (defaults to false)

/ trace : lists the last 100 HTTP request details

Extra for MVC projects;

/ docs : If spring-boot-actuator-docs is defined in classpath, it shows documents and sample request-results related to Actuators

/ heapdump : compressed hprof heap dump file

/ jolokia : If Jolokia is defined in classpath, it shows JMX beans over HTTP

/ logfile : Shows the current log file content if logging.file or logging.path values ​​are defined

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

Spring MVC is a complete HTTP oriented MVC framework managed by Spring Framework and centered on Servlets. It is equivalent to JSF in the JavaEE stack. The most popular items in @Controller are annotated classes where you implement methods that you can access using different HTTP requests. Equivalent to implementing @RestControllerREST based APIs.

Spring boot is a tool to quickly install applications, it offers out of the box configuration for building Spring powered applications. As you know, Spring integrates a wide variety of modules under its umbrella such as spring kernel, spring data, spring network (which includes Spring MVC by the way) and so on. With this tool you can tell Spring how many to use and you'll get a quick setup for them (you'll then be allowed to change them yourself).

So, Spring MVC is a framework to be used in web applications and Spring Boot is a production ready project starter based on Spring.

# Why to use VCS ?

If you want to preserve different versions of a file, it is wise to use a Version Control System (VCS). VCS allows you to access a specific historical version of files or the entire project, compare changes made over time, see who last made changes to what caused the issue, who introduced a specific bug, when, and much more. On the other hand, using VCS helps you easily make up for it when you make a mistake or accidentally delete some files. Moreover, all this does not impose any significant overhead on you.

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

## SOLID Principles

1. Single Responsibility: Our classes should have a single well-defined responsibility.
2. Open/Closed : Our classes are closed to change but should be open to adding new behaviors.
3. Liskov Substitution : We should be able to use the derived classes (sub class) instead of the parent class (base class) from which they are derived, without making any changes to our code.
4. Interface Segregation: We should prefer to create more specialized contracts instead of a single contract for general use.
5. Dependency Inversion: In layered architectures, top-level modules should not be directly dependent on lower-level modules.

## Example

*Person.java*

*public class Person*

*{*

*private Long personId;*

*private String firstName;*

*private String lastName;*

*private String age;*

*private List<Account> accounts;*

*}*

*Account.java*

*public class Account*

*{*

*private Long guid;*

*private String accountNumber;*

*private String accountName;*

*private String status;*

*private String type;*

*}*

# What is RAD model ?

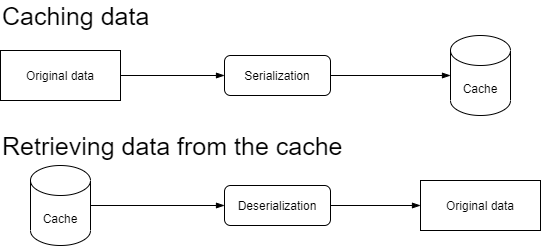
Rapid Application Development (RAD) is a set of methods applied to software development processes and supporting the realization of these processes as quickly as possible. The main objectives of this method, which supports the rapid completion of processes; can be explained as high speed, high quality and low cost. This methodology, which aims to increase the quality of software development processes and reduce costs, is an up-to-date method as it also saves time.

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

Starters are simply a set of dependency descriptors that you can add to your application. It saves you the trouble of searching for each of the technologies you want to use and adding them as dependencies one by one. Thanks to Starters, you can easily add Spring and related technologies you need to your application. If you want to use Spring and JPA as an example, it will be sufficient to add the "spring-boot-starter-data-jpa" dependency to your project.

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

Cache is one of the techniques we use to improve the performance of our application. Let's make things clear by giving an example. For example, we are throwing a query to the database. We first write the incoming data to the cache. Then we show it to the user. If the user runs the same query later, it serves the data from the cache instead of going to the database. Another example we can give is algorithms with long processing times. For example, we called an algorithm with parameters param1=4, param2=5, param3=10. After taking a minute, it gave the result 123. If we call the algorithm again with the same data (param1=4, param2=5, param3=10) it will give us the result 123 in less than a second. 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 a few annotations we need to know before we can do cache operations.

**@Cacheable(“cacheName”)** : If the relevant key value (cacheName) has not been written before, it will be cached first. Then the relevant values ​​are returned to the user. If there is relevant information in the cache, it will be called from the cache. For example, if @Cacheable(“movies”) is written above a method, it will come from the database the first time that method is called. The second time it is called, it will come from the cache.

**@CacheEvict(“cacheName”)** : If there is a value written with the corresponding key in the cache, it will delete it. For example, we called a method called @Cacheable(“movies”) and it cached the movie list. If we then run the method with the expression @CacheEvict(“movies”), it will clear the movie list in memory.

**@CachePut(“cacheName”)** : If we want a method to run every time, we use the method with this annotation. This way, the cache will always have the most up-to-date data. For example, when the method written @CachePut(“movies”) is executed, it goes to the database and caches the most recent information in the database. If the method with the Cacheable(“movies”) annotation is called later, it will give the most up-to-date information.

# What & How & Where & Why to logging ?

In the computer world, a log is a file that is automatically generated when certain events occur on your system. Log files are often timestamped and can record practically everything that happens behind the scenes in operating systems or software applications. In short; it records anything it considers important to monitor, whether on the server, within the network, operating system, or application. Logs can document any event, from messages and transactions that occur between different users, to those that occur during a backup, to errors that prevent an application from running, or to files requested by users from a website.

There are several types of logs. Some are human-readable, while others are held for auditing purposes and are not human-readable. Audit logs, transaction logs, event logs, error logs, message logs are just a few examples of different log files, each serving a different purpose.

It comes in a wide variety of extensions such as .log, .txt or different proprietary extensions. Depending on the extension and readability, it can be opened with a standard text editor (such as Notepad) or a word processing application (OpenOffice or Microsoft Word) or may require certain applications to be opened.

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

One of the most important needs in Web API development is the need for documentation. Because what the API methods do and how they are used should be clear in the documentation. The reason for this is that the higher the readability of the created APIs, the easier it can be for the teams to use the API.

Swagger supports many languages.

An important purpose of Swagger is to provide an interface for RestAPI.