Summary

IOC: IOC stands for inversion of control. It is a principle in software which transfers the control of object or portions of program to container or framework

DI: DI stands for dependency injection. Dependency injection means providing the objects that an object depend on instead of creating them itself. DI can be used to implement IOC where the operation of setting an object's dependencies is transferred to IOC container.

Pros and Cons for different dependency injection approaches:

Constructor based:

Pros:

1. It guarantee the order of injections and ruled out the possibility of circular dependency.
2. It can be used for FINAL field.
3. It makes unit test easier.

Cons:

It can look redundant with less readability and difficult to maintain.

Setter based:

Pros:

1. It allows partial injection.
2. It improve performance as the injection happens only when the object is needed.

Cons:

1. It can have security issue due to reflection.
2. It cannot be used for object declared by final keyword.

Field based:

Pros:

1. It is easy to use and maintain and it has good readability
2. Circular dependency is not a problem

Cons:

1. It cannot be used for object declared by final keyword.
2. It is highly coupled with container. We cannot use it without IOC container.

Bean scope: Bean scope controls the lifecycle aspect of Bean such as the time of instantiation, the duration of Bean to be alive and the number of objects to be created throughout.

Bean Lifecycle: Bean life cycle refers to when and how the bean is instantiated, what action it performs until it lives, and when and how it is destroyed. It mainly follows the route of bean instantiation, dependency injection, custom init() method call, custom utility method use, and custom destroy() method upon destroying.

AOP: AOP stands for aspect-oriented programming. It entails breaking down program logic into distinct parts called concerns. The functions that deal with these concerns can be separated from the business logic and cut across multiple points in the business logic whenever needed.

Aspect: an aspect is a class that implements application concerns that cut across multiple classes

Advice: denotes the action to be taken by an aspect at a particular join point where actions include before, after, after return, after throw and around.

Joinpoint: Joinpoint is a point in. the program execution of the application where an aspect can be inserted into.

PointCut: Pointcut defines at what joinpoints, the associated advice should be applied.

Target: Targets are objects on which advice are applied.

@Transactional: @Transactional is an annotation that allows spring to execute transactions in application layer. Method declared by @Transactional will be transaction manager which coordinate other transactions to join and forms a single transaction.

SpringMVC: is a web framework that build upon Spring to facilitate web development

MVC: MVC stands for model, view, controller, and it is a design paradigm that guide the construct a web application. It consist of models that containing data, controller that take request and perform business logic and view that display information.

Spring MVC workflow:
First, requests are intercepted by DispatcherServlet. The DispatcherServlet loopup the handler mapping to find the corresponding controller to be interacted with. After communicated with controller, it gets data that are bind to ModelAndView and it send to ViewResolver to process. Once it received the processed view it will respond back to the request.

SpringBoot: SpringBoot is build upon Spring framework to create a lightweight development environment.

SpringBoot advantages:
1. It provides a flexible way to configure java beans, xml configuration, and database transaction
2. It provides a powerful batch processing and manages rest endpoints
3. Default configuration is automatic embedded out of the box
4. It offers easy-to-use annotation to ease the application development
5. It makes dependency management easier
6. Tomcat is embedded to SpringBoot and can be used out of the box

SpringBoot starter:

Spring Boot starter provides an easy way to handle dependency management. It contains all necessary packages that need to accommodation to each other with correct versions out of the box.

Auto Configuration: By using @SpringBootApplication, it entails the project is automatically configured with dependencies to be managed by spring boot.

REST API design: It is about the design strategy for REST API. In general, a REST API should be designed such that it uses http methods to indicate what kind of operation a client want to take and uses http url to indicate which service and data a client want to use and what kind of data they request.

Spring Restful API: Spring MVC provides multiple annotations that support the REST API design such as RequestMapping, Controller/RestController, etc.

Exception handling process: It is a process of how the application handle an exception. It generally involves checking @ExceptionHandler method, @ControllerAdvice class and @ResponseStatus attached to exception. Following a cascading rule, the application will handle exceptions accordingly.

Validation: Validation is a process of checking whether a specific field matches what is required by the application on the backend. The application will not allow invalid operation to be taken and will sent corresponding response to client.

Swagger: Swagger is a java package that makes the communication easier between backend and frontend as it provide the frontend developer the endpoint to visualize backend API without having them look into backend sourcecode.