Spring

Solves the problem of dependency injection.

build loosely coupled applications. It can be easily unit tested.

Duplication / Plumbing Code.

Good Integration with other frameworks.

Spring MVC

concerned with developing web application in simple way.

Spring Boot

Instead of configuring everything → it adds spring MVC jar and auto configuration is done

JSON
Tomcat
Spring Framework
Log in frameworks

} Spring boot starter contai

Yaml → language that Spring boot understands.

Spring boot starter web :
web application
" " starter Test -
Unit Test

JPA ⇒ interface for hypernet

[includes lot more maven dependecies

spring boot starter - web - services - SOAP webservices

- REST services

" jdbc - for jdbc

" hateoas - for hateoas

" security - for security

" cache

~~Caching~~ - caching.

Actuator:

Spring AOP:

Aspect Oriented Programming.

@Service — Business class

@Repository → Dao (Data class)

@Aspect → AOP

@Configuration → Configuration.

@Before(→ Before execution.

Join Point → to know the details.

Advice: What should I do when intercepted

pointcut: Methods to be Intercepted

Aspect : combination of Advice and the pointcut

Joinpoint : specific interception of a method call.

} Weaving process.

Weaving - Process of implementing the AOP around the method call

Weaver - the Framework which implement weaving

@ AfterReturning → used after execution.

@ AfterThrowing → when a method ~~sho~~ throws an exception.

@ Around.

@ Target →

Important features of Spring Framework:

→ dependency injection

→ creating loosely coupled applications

Spring initializer → start.spring.io

Dependency:

One class depends on another class that without which another class can't be working.

Tight Coupling:

This kind of code will be complex.

classes are directly instantiating.

Loosely coupling:

→ It replaces the directly instantiating by using the constructor.

→ We are not creating an instance of class directly here

→ we are creating a constructor.

Spring Framework:

User needs to tell

→ What are the objects it needs to manage?

→ What are the dependencies of your particular class?

Annotation:

@ Component: Spring start managing instances of the present class and dependency class.

@Autowired → helps to tell the dependency class is depends by the complex business service class.

Beans:

Different objects that are managed by the spring framework.

AutoWiring:

The process where spring identifies the dependences, identifies the matches for the dependencies and populates them.

Inversion of Control:

Taking the control from the class that needs the dependency and giving the control to the framework to the spring framework.

Ioc Container

Inversion of control Container

→ Ioc Container is a generic terminology to represent anything that implements the inversion of control.

Application Context

→ where all the beans are created and managed.

→ Core Logic of the spring framework happens.

To create a spring project

→ start.spring.io → download a spring file.

→ file → import → existing maven project → select pom file → finish

→ file will be created

IOC Container:

Responsible for create, coine, configure and manage objects during their complete life cycle.

Uses metadata Configuration

Types of Spring IOC Containes:

1. Bean Factory
2. Application Context.

Bean Factory:

Simple container which provides the basic support for dependency injection

Application Context:

It includes all functionalities of the BeanFactory container with some extra functionality like internationalization, even listenners etc.

Spring framecoork bean scopes:

1. Singleton: Spring Container keeps it into a cache and returns the same instance each time a request for that particular bean is made

2. prototype: It scopes a single bean definition to have neco bean instance each time a request for that particula bean is made.

3. request: It scopes a bean defintion to a HTTP request

4 session: HTTP session

5. global-session: to globall HTTP session.

Injection :

It is a process of passing the dependency to a dependent object

Types of dependency Injection:

1. Constructor-based Dependency Injection
2. Setter-based Dependency Injection.

Autowiring :

It is the process of placing an instance of one bean into the specified field in an instance of another bean.

Autowiring modes:

1. no.
2. byName
3. byType
4. Constructor.
5. auto detect.

Autowire By name:

This will inspect the application context and look for a bean named exactly the same as the property which needs to be autowired.

Problem with JDBC API:

We have to write unnecessary code for creating database connections, SQL statement executions, exception handling transaction handling and closing database connections etc.

Spring JDBC framework:

In case of spring JDBC framework it takes care of all things like creating database connections, executing SQL

statements, handling exceptions, handling transactions and closing database connections etc.

Annotation:

@ Required:
→ to enforce a required property.

@ Qualifier:
to avoid confusion which occurs when you create more than one bean of same type and want to wire only one of them property.

@ Configuration.
→ it is configuration using Java class
→ it is an analog for XML configuration.
→ used on class which defines beans.

@ Component Scan:
→ allow spring to know the packages to scan for annotated components.

@ Bean
→ used at method level
→ to create Spring beans.

@ Value:
→ indicates a default value expressions for the field or parameter to initialize the property with.

Stereotype Annotations:

@ Component → to add component scanning mechanism.

@ Controller → Used to indicate the class is a Spring controller.

@ Service → marks a Java class that performs some service.

@ Repository → to access the database.

Spring Boot Annotations:

@ Enable Auto Configurations : defines a base "search package".
→ tells the spring boot to start adding beans on classpath settings, other beans and various property settings.

@ Spring Boot Applications

Spring MVC and REST Annotations

@ Request Mapping:
→ used in both method and class level:
→ annotations is used to map web requests onto specific handler classes and handler methods.

@ CrossOrigin
→ enables cross origin requests

Composed @ Request Mapping Variants

@ Get Mapping
HTTP GET

@PostMapping
HTTP POST

@PutMapping → HTTP PUT

@PatchMapping → HTTP PATCH

@DeleteMapping → HTTP DELETE

@ExceptionHandler → handle exceptions

@Mappings and @Mapping :
indicates a web mapping annotations

@MatrixVariable

@PathVariable: