**Spring**

* **Spring framework-**

1. Before Spring we used to use struct.
2. Spring is a java based framework.
3. Spring is a non-invasive framework which means we don't need to implements and extends spring related classes and interfaces for operation.
4. Spring is a versatile framework.

5. By using spring we can develop several applications :

1) Web applications (Website)

2) Standalone application (Desktop based application which is non sharable - which can run without internate. eg. Notepad, Paint)

3) Distributed application :

If one application is communicating with another application for business purpose is called as distributed systems.

Eg🡪 amazon uses phone pay, google pay, paytm for payment purpose. This two are different application communicating with each other for business purpose

4) Microservice based application - Design Pattern

1. Spring provides several modules

total 21+ modules are available

To create the distributed web application we need to use these 7 modules

The 7 main core modules are :

1) Spring Core

Spring core provides you dependency injection and IOC-Inversion of control

2) Spring AOP – (Aspect oriented programming)

It is used to differenciate primary logic and secondary logic of our application

3) Spring Context -

It is used to create configuration related files

4) Spring DAO – (Data Access object)

It is used to provide persistent layer (DB layer) in our application.

5) Spring Web MVC –(Model View Controller)

It is used to create web application.

6) Spring ORM – (Object Relational Mapping)

It is used to provide persistent layer with object relational mapping

7) Spring Data JPA- (Java Persistence API)

It is used to provide persistent layer with JPA related operations

8) Spring Batch

9) Spring Cloud

10) Spring Security

* **In Spring we call Objects -> Beans**

**A diagram of a student

AI-generated content may be incorrect.**

**Questions-**

1. **What is container?**

* **It is the core component responsible for managing the lifecycle, configuration, and dependencies of beans (objects). It is also known as the Spring IoC (Inversion of Control) container.**

1. **What is beans.xml?**
2. **Diff. between core container & j2ee container (application context)**A white box with black text

   AI-generated content may be incorrect.
3. **What is bean tag?**

* **IOC-**
* It is use to provide dependency injection.
* It is use to manage beans of our application.
* Ioc contains diff. containers

1. Core container
2. J2ee container (application context)

**Container-**

It is use to create bean.

It is use to manage bean life cycle.

It is use to read bean config file & after finishing work of bean it will destroy it.

**Dependency Injection-**

It is the process of injecting 1 class bean into another class bean

we can perforn dependency injection using 2 ways

1. Setter getter based injection-

If we set variables, injecting 1 object into another using setter getter it is called as setter getter based injection.

In beans config file we can set, inject values by using <property> tag

1. Contructor based injection-

If we set parameters, injecting values into another classes using parametrized constructor it is called as contructor based injection.

In beans config file we can set variables, injects values we can use <constructor-arg> tag

If we use parameterized constructor and setter getter in class then setter getter always override the values.

**Beans Scopes-**

Scope is use to define behaviour of beans (how many objects you want to create you can)

There are multiple types of scopes.

1. **IOC-**

Singleton- whenever we call get bean method it will provide single object 🡪(it is default )

Prototype- whenever we call it it will create diff objects 🡪(we have to use scope=”prototype” in bean).

* If we make prototype to only parent class bean then child class will use default singleton. so hashcodes of parents are different(creating new objects every time) but child hashcode will remain same (because of singleton)
* If parent class bean is singleton & child class bean is prototype then it will not checks childs scope. So hashcode of parent is same & childs hascode also remain same.

In that case we can use <lookup-method name="get --that child class--"/>

1. **WEBMVC-**

Request- for every http request call it create new bean

Session- for every http session it create new bean.

**Wiring-**

Injecting 1 bean into another it is called wiring.

Earlier we perform dependency injection manually but in real time we have more than 1 beans so we need to write injection property every time which will increase your broiler plate code. For over coming this problem spring provide you autowiring.

**Autowiring-**

It provides you automatic injection based on multiple modes. Doing it automatically is called as autowiring.

It supports only non-primitive datatypes.

1. ***Bytype🡪*** it will check property datatype

When we use bytype & if we have multiple beans having same type then it will give you ambiguity. To overcome this problem we use byname.

1. ***Byname🡪*** it will check propertys name which is declare in parent pojoclass dependencyinjection variable name.

Based on that values it will search respected bean inside your beans.xml file

1. ***Constructor🡪*** if we perform automatic injection by using <constructor-arg> in that case we use constructor. It first checks bytype then byname.
2. ***Autodetect🡪*** it internally calls constructor

Q. bean lifecycle?

🡪 A screenshot of a computer program

AI-generated content may be incorrect.

**Annotation based configuration-**

1. ***@Configuration-*** it is the class level annotation which is used to represent java class as a spring bean config. file.
2. ***@Bean-*** it is used to represent configuration method as spring bean.
3. ***@Scope-*** it used to define scopes of the bean.
4. ***@Autowired-*** it is used to perform automatic injection by using diff. modes. We can use autowired annotation on field, constructor, method. It is also called as field level injection.
5. ***@Qualifier-*** it is use to define qualified name to a bean.
6. ***@Primary-*** if we have multiple beans having same type in that case if we want to represent a particular bean as a default bean we can use primary annotation.

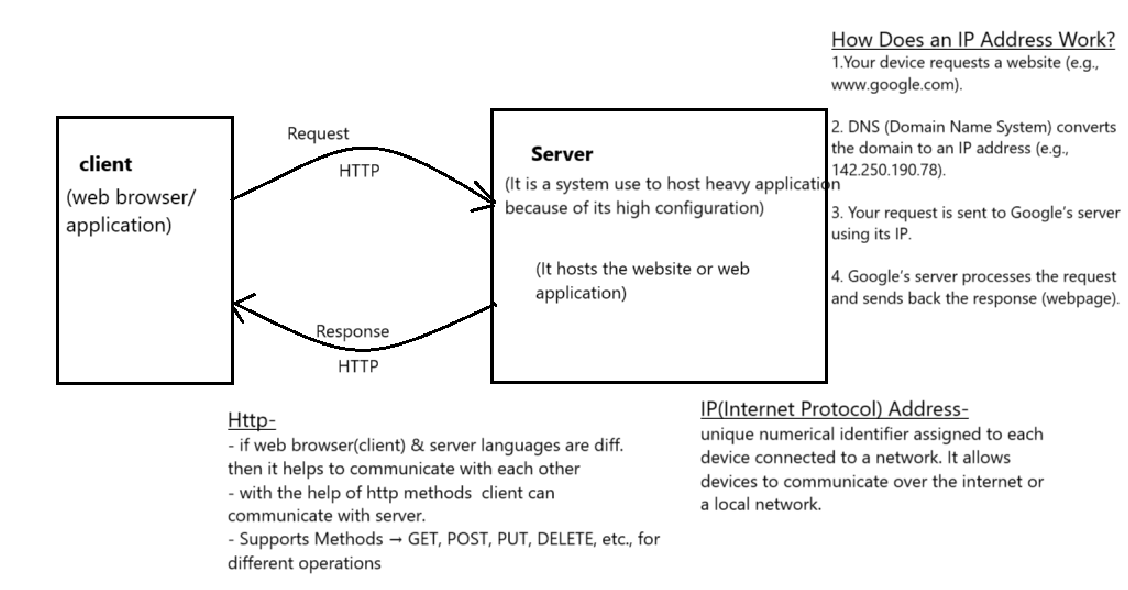
It used to marked bean as a default bean.

1. ***@PostConstruct-*** it is used to define custom init() method.
2. ***@PreDestroy-*** it is use to define custom destroy() method.

**Q. how to create custom init() method & destroy() method without implementing spring inbuild interface methods?**

**Q. Diff between Spring & Servlet bean life cycle?**

* **Spring Web MVC-**

****

**A black screen with white text

AI-generated content may be incorrect.**

**A black screen with white text

AI-generated content may be incorrect.**