**Spring IOC**

* **Spring Frameworks:-**
* Before Spring we used to use struct.
* Spring is a java based framework.
* Spring is a non-invasive framework.
* (which means we don't need to implements and extends spring related classes and interfaces for operation.)
* Spring is a versatile framework.
* (Which means any java framework use for develop application)
* Lightweight framework.

By using spring we can develop several applications :

1) **Web applications (Website)**

2) **Standalone application** (Desktop based application - which can run without internet (eg. Notepad, Paint)

3) **Distributed application :**

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

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

* **Spring provides several modules:-**

total 21+ modules are available

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

**The seven 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 differentiate 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:-**

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**

**Internal of springs?**

**A diagram of a student

AI-generated content may be incorrect.**

**What is container?**

**What is beans.xml?**

**Difference between core container and j2ee container(application context)**

|  |  |
| --- | --- |
| **Bean factory (I)** | **j2ee container(application context) (I)** |
| 1.core container | Internally calls beanfactory |
| 2.It does not xml configuration file directly it require resource | It directly reads xml config file. |
| .It reads at a time single xml config file | It reads at a time multiple xml config file |
| Lazy loader | Eager Loader |
| It supports only xml | It supports xml with annotation based |
| Parent Container | BeanFactory is child of application context |

**Q. What is bean tag?**

* **Beans means object of class.**

**Q. What is IOC?**

* Pre-defined program/design principal design
* Main purpose of IOC managing beans in spring program.
* IOC helps using container. Container managing beans life cycle.

**Q. 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.

**What is use of DTD?**

What is j2

What is eager loader

First you can create Application context object then it will be by default create bean object

What is lazy loader

* **Why is dependency injection?(OR manual wiring / auto wiring)**
* It is a process of inject in one class bean into another class. (means aggregation in java lang)
* **bean we can perform dependency injection using two way:-**

**1.setter getter based injection:-**

* if we set variable ,inject in one object into another object into another using setter getter it is called as setter getter based injection.
* In beans configuration files we can set, inject values by using <property> tag

**2.parametorized constructor based injection:-**

* If we set variable , inject in values into another classes using parameterized constructor it is called as constructor based inject
* In beans configuration file we can set variable, inject values using <constructor-arg> tag
* **Beans scope :-**
* Scopes is used to defining behavior of beans (how many object you want to create)
* There are multiple types of scope

1. **IOC- singleton ,prototype:-**
2. **singleton:-** Every time whenever we call get bean method single object(it is default).
3. **prototype:-** Every time whenever we call get bean method different object. (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, session.**
2. **request:-** For every http request call it will create new bean.
3. **session:-** For every http session it will create new bean.

* **IOC (INVERSION OF CONTROL):-**
* It is used to provide dependency injection
* It is used to manage beans of our application
* IOC contain different container

1. **Core container(Bean factory)**
2. **J2ee container(Application context)**

* **Container:-**
* Container it is used create bean
* It is used to manage bean life cycle
* It is used to read beans configuration file and after finishing work of bean it will destroy bean
* **Wiring:-** inject 1 bean to another it is called wiring

Earlier we perform dependency perform but in real time if we have more than 1 bean when to write injection property for every bean which will your boiler plate code for overcoming problem spring provides you auto wiring .

Auto wiring provides you automatic injection based on multiple modes

**Note:-** autowiring supports only non-primitives data type.

* **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 (class name)

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 property name (class field name)

* Based on that value it will search respected bean insider you beans.xml file
* When we used by type and if we have multiple beans having same type then it will give you ambiguity to overcome this problem we will ” byname”

1. **Constructor :-** If we perform automatic injection using< constructor-arg> tag in that case we use constructor. Constructor first check “bytype” then “byname”
2. **Autodetect:-** autodetect internally calls constructor. And Constructor internally call by type and by name

**Q.**

A screenshot of a computer program

AI-generated content may be incorrect.

**JAVA BASE ANNOTATION**

1. **@Configuration:-** It is a class level configuration it Is used to represent java class a spring bean configuration file.

@configuration It is provide multiple bean our requirement.

1. **@Bean:-** it is used to represent configuration method as spring bean.

It is method level annotation.

1. **@Scope:-** It is used to defines scope of bean.
2. **@autowired:-** it is used to perform automatic injection by using different modes. We can use autowired annotation on field, constructor, method. It is also called as field level injection.
3. **@Qualifier:-** It is used to define qualified name to a bean.
4. **@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 is used to mark bean as a default bean).
5. **@PostConstruct:-** It is used to define custom init method.
6. **@PreConstruct:-** It is used to define custom destroy method.

**Q. Difference between spring and servlet Bean life cycle?**

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