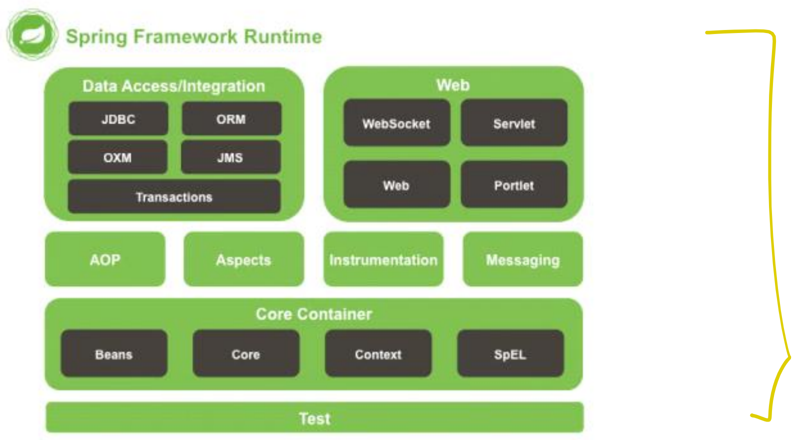
Spring core

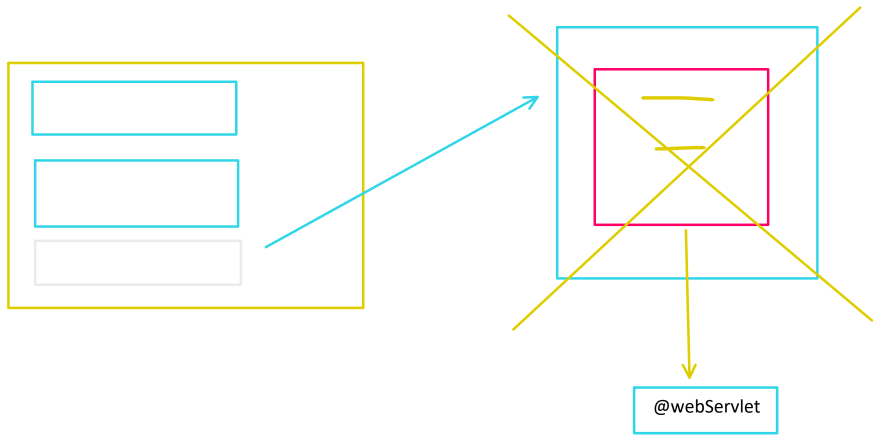
20 July 2025 09:03

Spring framework



Framework:

1. semi developed software which provides common logic required for project development
2. Framework will help developers to implement more functionality in less time
3. When we use framework to develop project, we can focus only on business logic



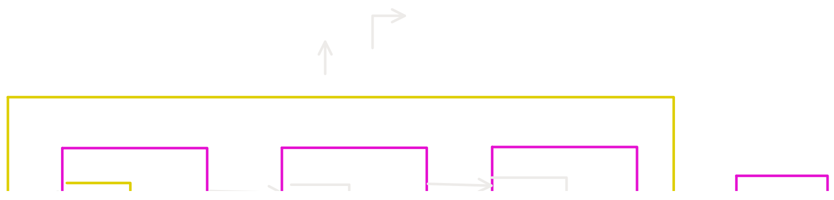
Type of frameworks:
----

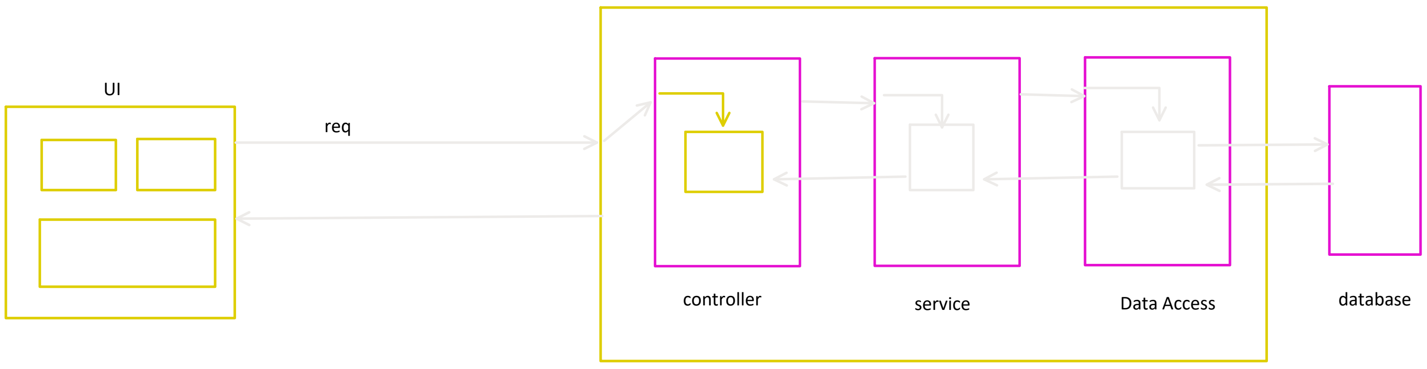
1. Frontend framework : Angular
2. Web framework : Struct (Outdated)
3. ORM framework : Hibernate -> Spring data JPA

By using struct we can develop only web layer in the project (Controllers)
By using JPA we develop only Data Access layer (persistence layer)

Architecture Application

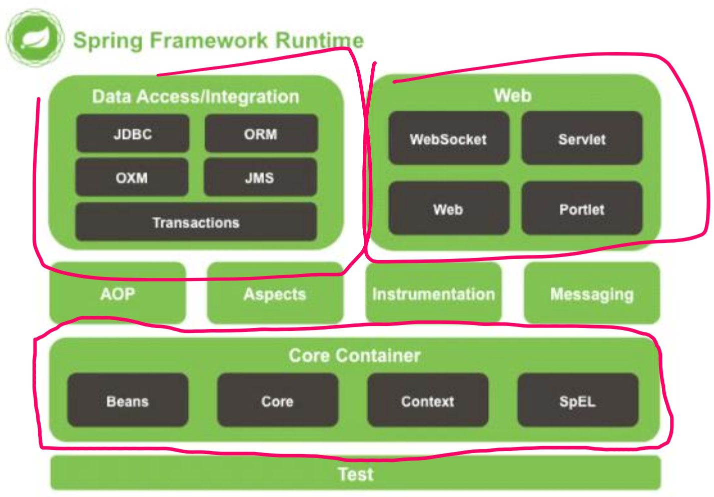
UI

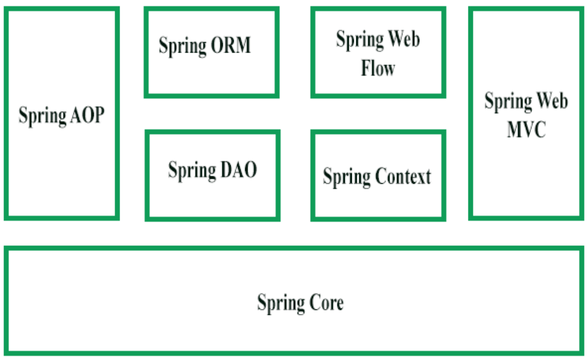




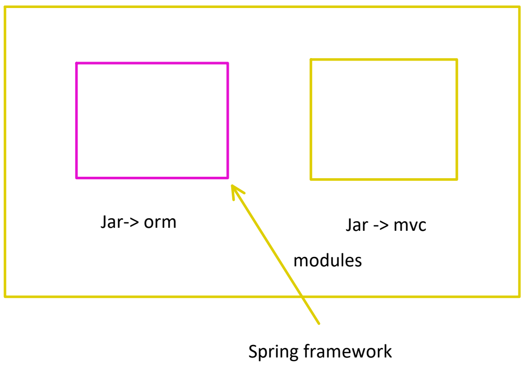
Note: to overcome the problem of struct framework, spring framework came into market

1. Spring f/w is called as application development framework
2. By using spring f/w we can develop end to end application
3. Spring f/w is free and open source
4. Spring f/w developed in modular fashion because this modular development it is loosely coupled.





Note : we just need spring core knowledge to work on spring f/w



Spring Modules
----

1. Spring core
2. Spring Context
3. Spring JDBC
4. Spring ORM
5. Spring AOP
6. Spring Web MVC
7. Spring Security
8. Spring Batch
9. Spring Data JPA
10. Spring Rest
11. Spring Cloud
12. Spring AI

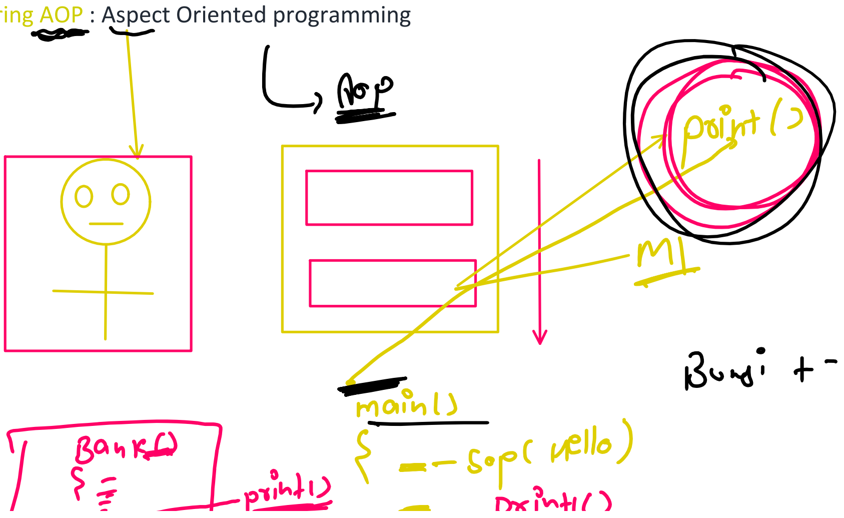
Note : Spring is very flexible framework. It will not force to use all modules. Based on requirement we can pick up particular module and we can use it.

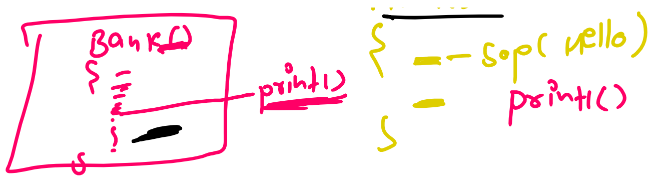
1. Spring is versatile framework (easily it can be integrated with other framework)
2. The current version of spring framework is 6.x
3. Spring f/w is under licences o VM ware Tanza
   a. URL : www.spring.io

1. Spring Core: It is base module in the spring f/w
   a. Spring core module providing fundamental concepts of spring f/w
      i. IOC container (Inversion of control)
      ii. Dependency injection
      iii. Bean life cycle
      iv. Bean Scope
      v. Autowiring etc….

2. Spring Context : it will deal with configuration required for our spring application.

3. Spring AOP : Aspect Oriented programming





Note: if we combine business logic & secondary logic then will face Maintenance issue of our project. So we use AOP to separate business logic and secondary logic.

4. Spring JDBC : Spring jdbc is used to simplify Database communication logic

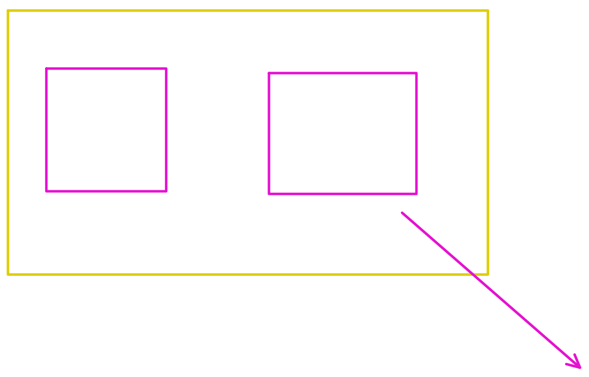
In java jdbc we need to write boiler plate code (repeated code) like below in several classes

1. LOAD DRIVER
2. GET CONNECTION
3. CRETAE CONNECTION
4. EXECUTE QUERY
5. CLOSE CONNECTION

=> Using Spring JDBC we can directly execute query the remaining part spring JDBC will take care

5. Spring Web MVC : it is used to developer both web Application & Distributed App
   => Web application: (C 2 B)
   Ex: Gmail.com, facebook.com

   => Distributed Application (B 2 B) / web services or restful Services
   q

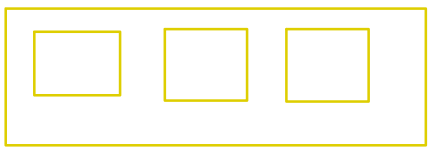


Ecomm : flpkart

Browser

Distributed app:
A distributed app is a system where components are spread across multiple nodes but work together as single system.

Zomato:



System

Teck stack :
Microservices
Kafka
Load balancer

Web application

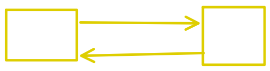
Distributed app

Web application

1 server
End user uses this via UI

Distributed app

Multiple server
Services talk to each other



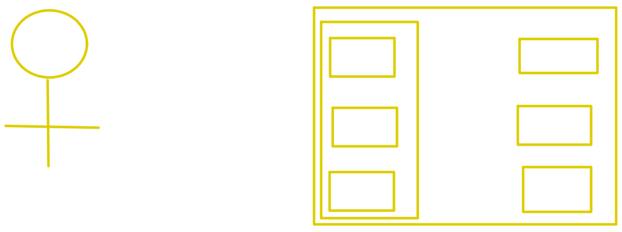
6. Spring ORM : Object relational mapping
   => spring f/w having integration with ORM frameworks
   Eg: Spring ORM, Spring Data JPA

Note: JDBC will represent data in text format where as hibernate ORM will represent data in Objects format

```
executeQuery(query)

While(rs.hastNext()){

Rs.getString(name)
Rs.getInt(id)
-----------------------------------------------------

Stundent  s = JPA.get():   => Object
```

7. Spring Security:
   a. Security is very important for every application
   b. Using Spring security we can implement Authentication & Authorization
   c. Spring security with Oauth 2.0
   d. Spring Security with JWT (JSON web token)

8. Spring Batch : Batch means bulk operation
   a. Reading data from excel and store it into database table
   b. Sending Monthly statement to customer in email
   c. Sending Reminders to customers as Bulk SMS.



10. Sprint Test : It provides unit test framework.

Note: Spring core -> it is all about managing dependencies among the classes with loose coupling

In project we will develop several classes. All those classes we can categorize into 3 types
1. POJO
2. Java Bean
3. Component

What is POJO ?

=> POJO : plan old java object
=> Any java class which can be compiled by using only JDK software is called a POJO class

Ex: 1
```
Class Demo {

	Int id ;
	String name ;
}
```

Ex 2:
```
Class Demo2 extends Thread {
	Int id ;
	String name ;
}
```

Ex 3:
```
Class Demo3 implements Runnable {

	Int id:
	String name;

	//run method
	Run()
	{
	}
}
```

Ex: 4 yes
```
Class Demo3 extends GenericServlet {

	Int id;
	String name;

	//run method
	Run()
	{
	}
	Service()
	{

	}
}
```

What is java beans

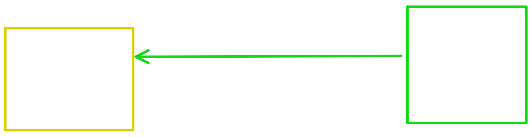
=> Any java class which follows beans specification rules is called as Java Bean.
1. Class should implement serializable interface
2. Class should have private data member (variables)
3. Every private variable should have public setter and public getter method
4. Class should have zero-param constructor

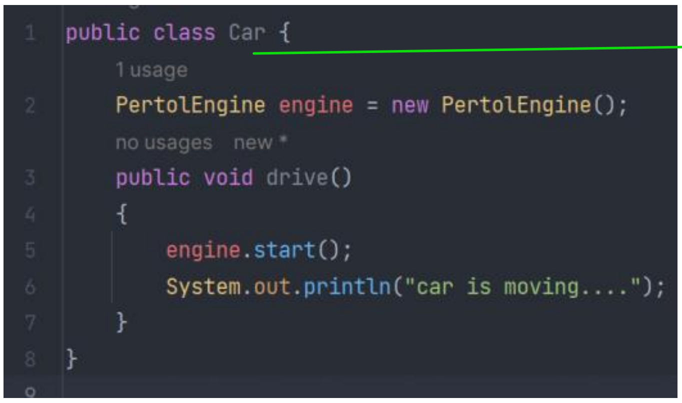
Note : Bean classes are used to write business logic and to store and retrieve data

What is component ?

The java classes which contains business logic is called as compnonet classes
	Eg: controller, service

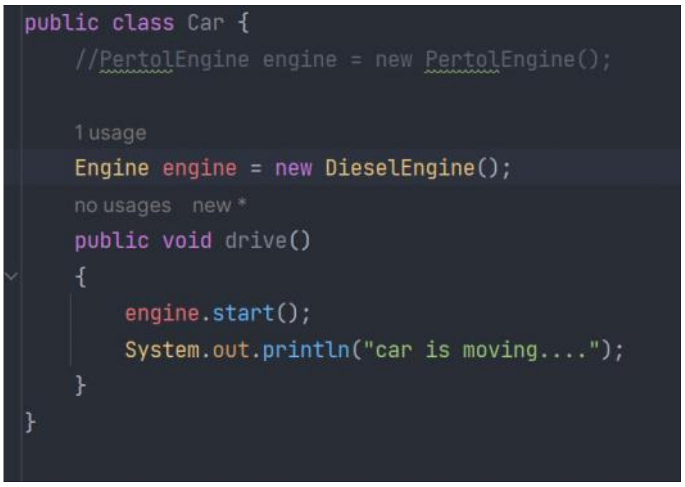
Tight Coupling





```java
public class Car {
    1 usage
    PertolEngine engine = new PertolEngine();
    no usages   new *
    public void drive()
    {
        engine.start();
        System.out.println("car is moving....");
    }
}
```

Car is tightly coupled with Engine class
We can't change engine type
We can't test the car without Engine
No flexibility



```java
public class Car {
    //PertolEngine engine = new PertolEngine();

    1 usage
    Engine engine = new DieselEngine();
    no usages   new *
    public void drive()
    {
        engine.start();
        System.out.println("car is moving....");
    }
}
```

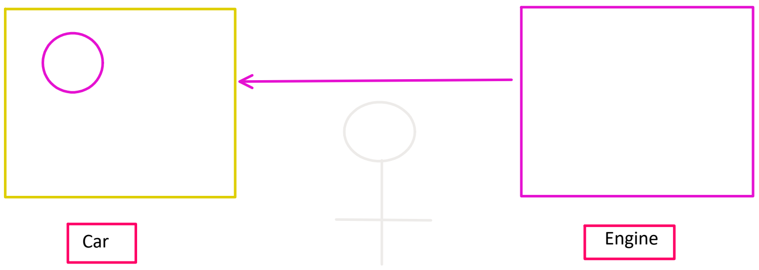
Flexible but still poor
We are using interface but still we have make object hardcoded new DieselEngine();

Without Factory

1. Car create engine
2. Tightly Coupled
3. Hard test
4. No Flexibility

With Factory

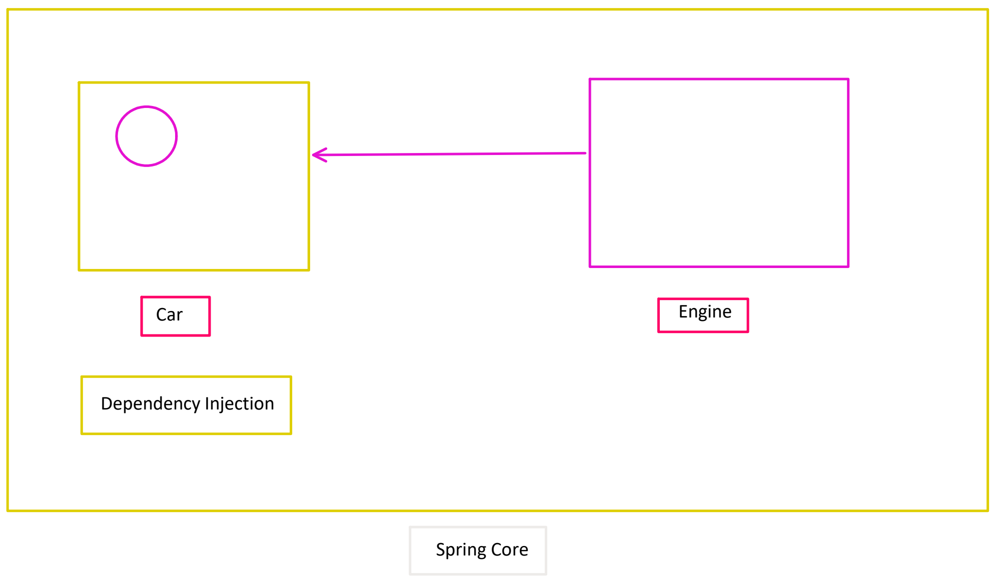
1. Engine is created by factory
2. Loosely Coupled
3. Easy test
4. Flexibility



Factory

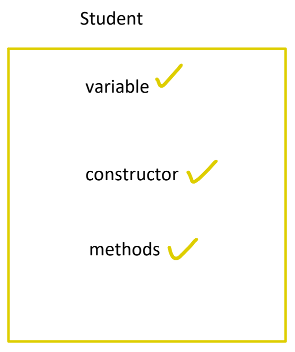
Dependency Injection

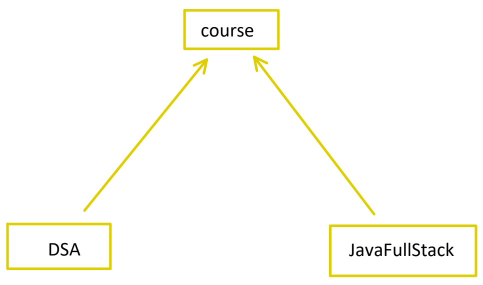
1. Loosely coupled
2. Dependency Injection

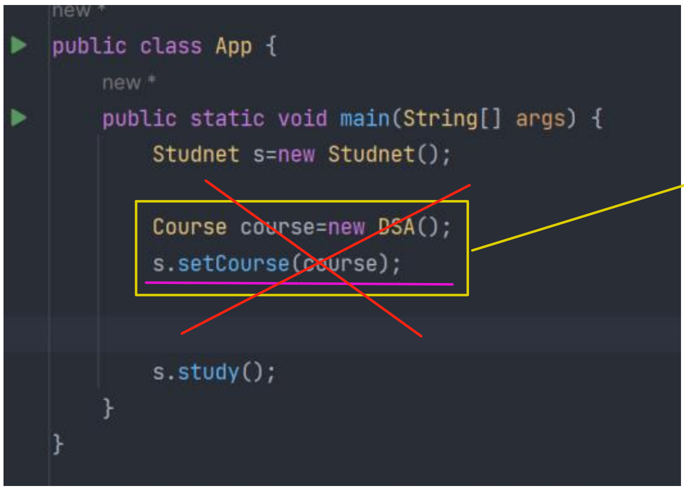


What is dependency injection ?

1. The process of injecting one class object into another class is called Dependency Injection.
2. We can perform dependency injection in 3 ways
   a. Setter injection
   b. Constructor injection
   c. Field injection





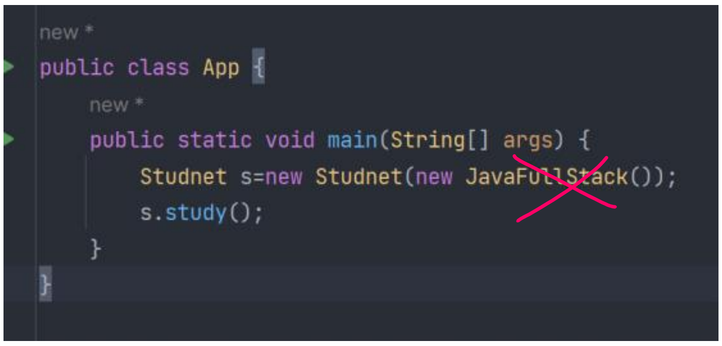


```
new *
public class App {
    new *
    public static void main(String[] args) {
        Studnet s=new Studnet();

        ~~Course course=new DSA();~~
        ~~s.setCourse(course);~~

        s.study();
    }
}
```

Dependency injection

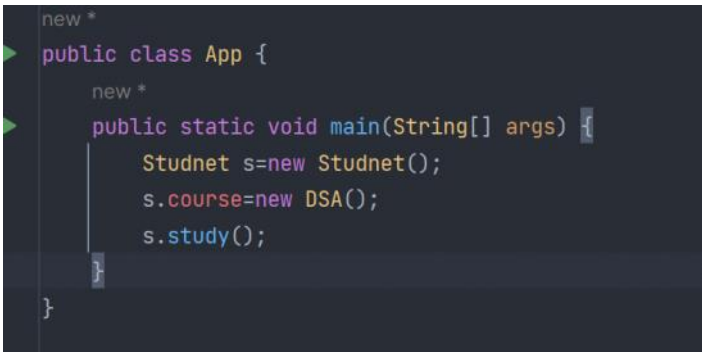
Manually injection

Constructor Injection



```
new *
public class App {
    new *
    public static void main(String[] args) {
        Studnet s=new Studnet(new ~~JavaFullStack~~());
        s.study();
    }
}
```

Field Injection



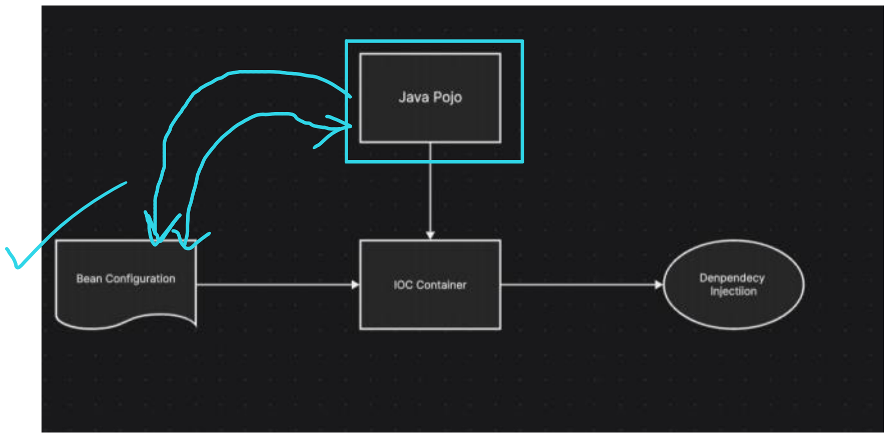
```
new *
public class App {
    new *
    public static void main(String[] args) {
        Studnet s=new Studnet();
        s.course=new DSA();
        s.study();
    }
}
```

Note: in manual injection we need to make variable public otherwise it will not available outside the class
But in spring core we don't have to make field public . Because spring uses reflection api to access private fields

What IOC container ?

1. IOC stands for Inversion of control
2. IOC is responsible for dependency injection in spring applications.
3. Dependency Injection means creating and injecting dependent bean object into target bean class

Note: IOC container will manage life cycle of spring beans.
Note: We need to provide " Java classes + Bean Configuration" as input for IOC then IOC will perform DI and provides spring beans which are ready to use.



What is Spring bean ?
----

1. Any java class whose lifecycle (creation to destruction) is managed by IOC is called spring bean.
2. We can represent java class as a spring bean in 2 ways
   a. XML approach : outdated
   b. Annotation approach : recommended

Note : In spring we can use both XML and annotation approaches. Spring boot will support only Annotations (no xml)

IOC Containers
----

1. BeanFactory (outdated)
2. ApplicationContext (recmm)

Ex : Application Context = new ClassPathApplicationContext(String configFile)

First Application Development using Spring core module

Pre-requisites: Jdk, IDE

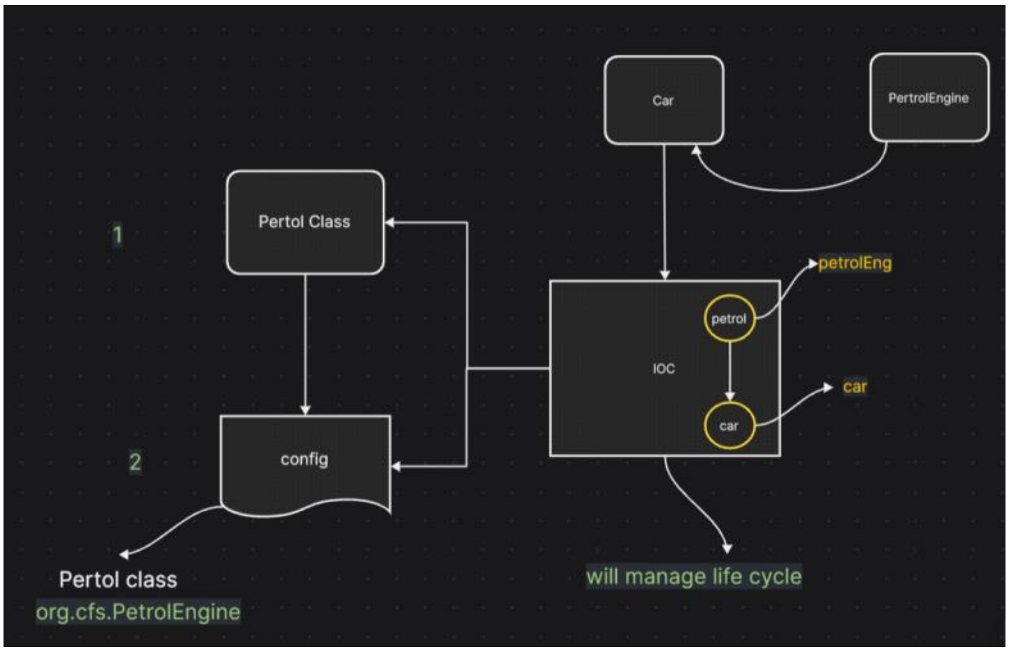
1. Create maven project in IDE
2. Add spring context dependency in pom.xml file

```
<!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
<dependency>
   <groupId>org.springframework</groupId>
   <artifactId>spring-context</artifactId>
   <version>6.2.9</version>
</dependency>
```

3. Create required java classes
4. Create bean configuration file and configure bean definations

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
                http://www.springframework.org/schema/beans/spring-beans.xsd">

</beans>
```

Internal how IOC works



Difference Between BeanFactory & ApplicationContext
----

1. BeanFactory will follow lazy loading concept that means when we request then only it will create bean object
2. ApplicationContext will follow eager loading

Note: any bean we IOC is creating is singleton object
Note: Spring bean default scope is singleton

Eager loading means creating objects for spring bean when IOC start
Lazy loading means creating objects for spring bean when we call getBean() method

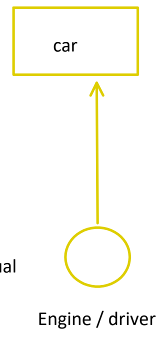
```
BeanFactory factory = new XmlBeanFactory(new ClassPathResource("your-
beans.xml"));
```
The XmlBeanFactory class in the Spring Framework was deprecated in Spring 3.1.

How to differentiate setter injection and constructor injection in Bean config file
----

1. <property> tag use for setter injection
2. <constructor-arg> use for constructor injection

Bean Scope:

Scope defines how many objects of spring bean are created within the container.
Type of scope:
1. Singleton
2. Prototype
3. Request
4. Session
5. Web socket



Real world example
----
1. Singleton -> One engine for all type of same car.
2. Prototype -> New engine for each car

Note : prototype beans are not automatically destroyed by spring if cleanup is needed , manual destruction is required.

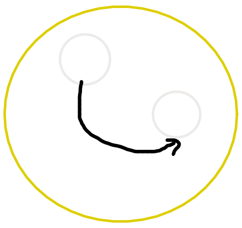
3. Request : for each http request we will create one object.
4. Session: for each session will create new object
5. Web socket : for each web socker will create new object

✓ Autowiring in Spring core

Autowiring is way to automatically inject dependencies into a bean without explicitly specifying <property name="engine" ref="petrolEng"> </property>

Note : it tells the IOC container please inject the suitable bean based on name, type, or cons parameter.

```
<bean id="petrolEng" class="org.cfs.PetrolEngine" scope="prototype"/>
<bean id="car" class="org.cfs.Car">
</bean>
```



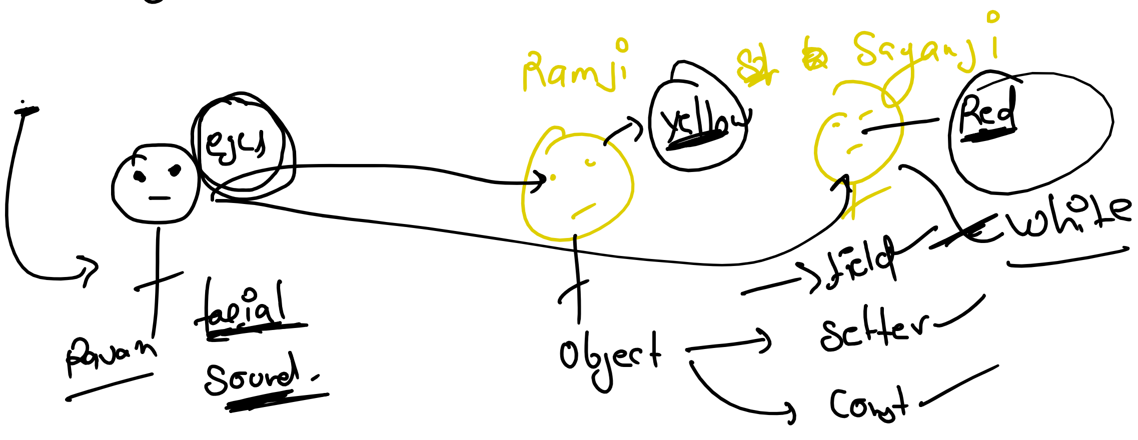
Modes of Autowiring

1.byName: Matches by property name (engine)
2.byType: Macthes by class/interface type (eg. Engine)
3.constructor : inject depedency via constructor
4.no (default) : no autowiring

Mode of Auto

1) byName →

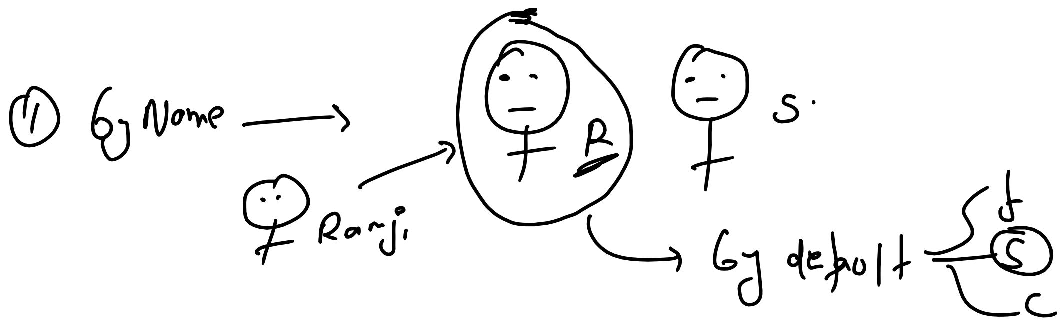
°ent → IOC



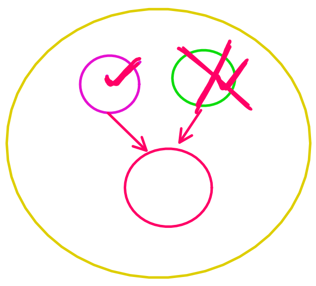
$\boxed{*}$ instruction

Modes → first

Object → ③ { f, c, s }



byType:



3. Constructor :

Sir raise byName me hum specify nhi krrhe property ka name , woh internally kaise penchan rha property name is engine

Spring Bean Life cycle

Spring bean life cycle = journey of spring bean from creation to destruction, managed by spring IOC container

Steps in life cycle:
1. Object creation
2. Property setting (DI)
3. Initialization
4. Business logic
5. destruction

Break → 30 $\boxed{10:30}$-25