**Spring Tutorial**

* Spring is a dependency injection framework to make java application loosely coupled. Spring provide IOC (Inversion of Container) for dependency injection.
* Spring Framework make easy development of JavaEE Application

**Dependency Injection –** It is a design pattern.

class X class Y

{ {

Y ob; public void doWork()

public void doWork() {

{

} }

} }

Here class X is dependent on class Y so this is what dependency. It makes our application loosely coupled. The object is created by spring.

* **IOC** – When we give control of object creation to spring. In this spring dynamically create the object at run time and inject that Y class object in class X. So this process is called IOC.

We have to provide metadata(information) what type of object we need. So we have two method by which we can provide information

1. By Xml file
2. By Annotation.

This whole process is used in J2EE application

**Spring MVC** **UI Layer ProductController**

| |

**Security** **Business/Service ProductService**

**Transaction management Layer |**

| **ProductDao**

**Spring jdbc** **Data Access layer-----------------------|** methods

**Spring ORM** |

**DB**

ProductDao dao = new PrdouctDao();

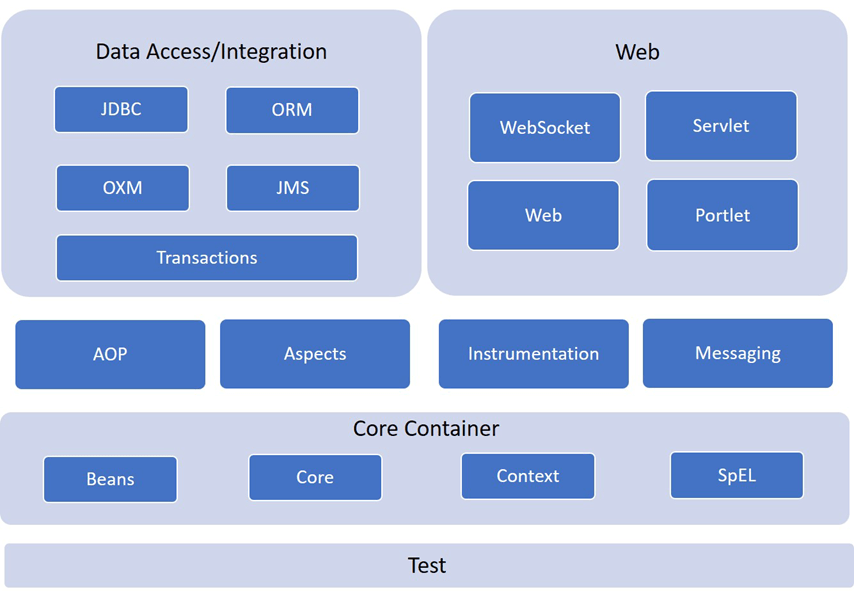
If we create object like this, then this application becomes tightly coupled.

So here we use dependency injection.

So, Spring create object of ProductDao and inject in ProductService and same for controller

With the help of metadata through xml or annotation we will tell what object we want in ProductService.

**Spring Modules**



**Core Container –**

* Core and bean provide us all the fundamental concept like IOC, Dependency injection etc.
* Context module inherit bean means all the features that bean have they also present in context and some other feature like internationlisation, resource loading etc.
* Spel is used to manipulate object at run time.

Next layer will provide us to decouple the code, so we distribute the functionality.

**Data Access layer –**

* Here we can integrate the tools for database like mysql for jdbc and hibernate for orm tools. And this provide us API with the help of this we can use database easily.

**Web Module –**

* This is used to integrate the web related project like Spring MVC.

**Spring IOC Container –**

* It is predefined programs. Used to manage the life cycle of object. This container is responsible for creation and destruction of object. With the help of this we have ready to use objects. With the help of xml we can describe what dependency are there on particular object

**Application Context(Interface) –**

* This represents our Srping IOC Container. We have three type of implementation for this application context.

1. ClasspathXmlApplicationContext – This scan our Xml Configuration from java class path.
2. AnnotationConfigApplicationcontext – This scan our Annotation Configuration.
3. FileSystemXmlApplication – This scan our configuration from file system.

**Ways Of injection**

class Student IOC Container class Address

{ {

int id; String street;

String name; String city;

Address address; }

}

Dependency injection can be done by 2 ways:

1. Using Setter Injection – For setting the values setter methods call.
2. Using Constructor Injection – In this constructor will call constructor to set the values.

* **Setter Injection –**

**IOC**

class Student class Address

{ {

id, name, address; street, city, state;

setId(id){} setStreet(street)

setName(name){} setCity(city)

setAddress(address){} setState(state)

} }

IOC will automatically call setter methods and set the values of street, city and state and save all these values in address object.

Same for student class when we create object of student class IOC will set the values of id, name, address (It takes values from address object).

* **Constructor Injection –**

class Student class Address

{ {

String id, name, address; String street, city, state;

Student (id, name, address) Address(street, city, state)

{ {

} }

} }

Now in this case IOC will automatically call constructor of address class and set the values. And then all these values we have in address object also now these values are ready to use. Same for student object.

* It is totally dependent on us which dependency injection we have to use to set values. We just simply mention in configuration file to use setter or constructor injection.

**Configuration File (XML File)**

* Where we declare beans (java class) and its dependency.

Type of data type (Dependenices) that we can inject:

1. Primitive DataTypes

Byte, short, char, int, float, double, long, Boolean

1. Collection type

List, set, Map and properties

1. Reference Type – User-defined data types.

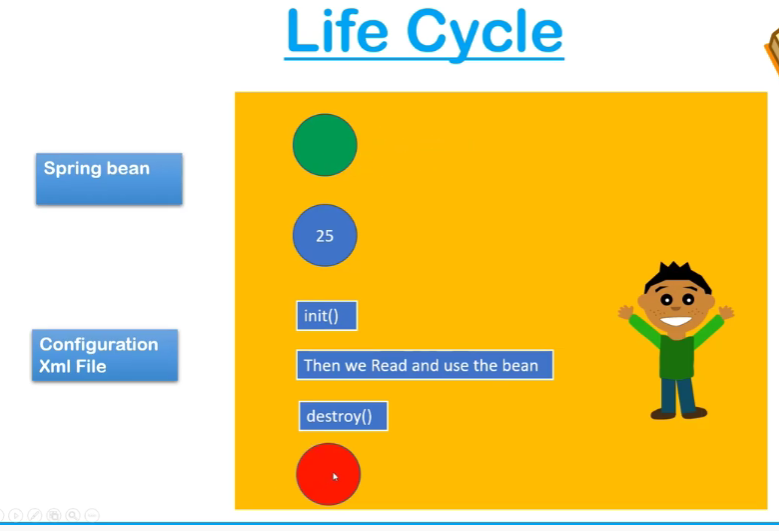
Other class object

**Life Cycle Of Spring Beans**

* Spring provide two important methods to every beans

1. public void init() – This is used to initialise value when we connect with database, configuration.
2. public void destroy() – It is used to clean up the code.

We can change the names of method. But signature must be same.

****

**Configuration Techniques**

* XML – Through xml file we can configure

public void init(), public void destroy()

* Spring Interface – Spring interface provides method which we can implement

InitializingBean() – This provide us init() only

disposable() – This provide us destroy()

* Annotation – Through annotation we can configure.

@PostConstruct – This provide init() functionality.

@PreDestroy – This provide destroy() functionality.

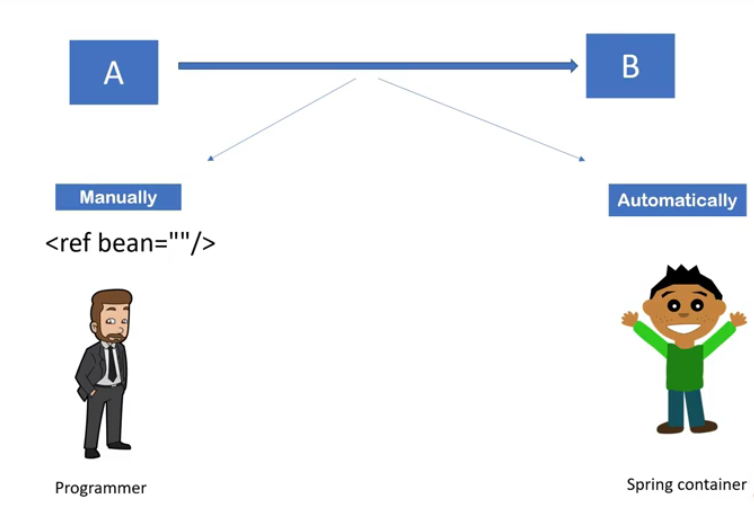
**Auto wiring in Spring**

* Feature of spring Framework in which spring container inject the dependency automatically.
* Autowiring can’t be used to inject primitive and string values. It works with reference only.

A --------------------------------------------------------------------🡪 B

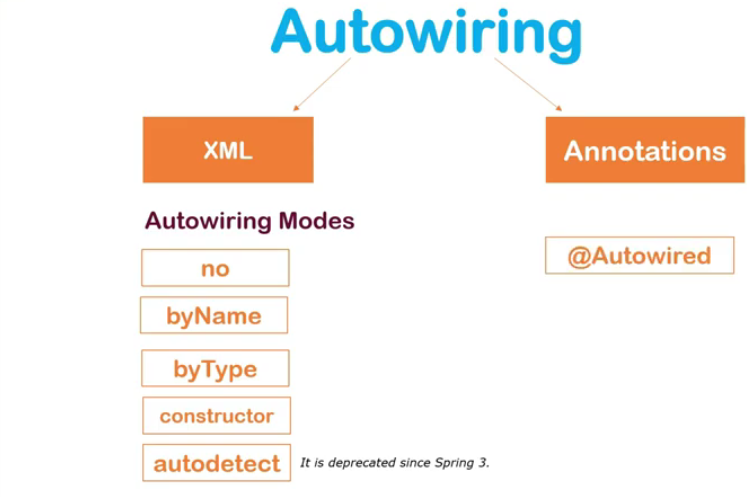
Here class A is totally dependent on B. When we inject class B object in class A then only class A can perform their operation. When we link these two objects then it is called wiring.

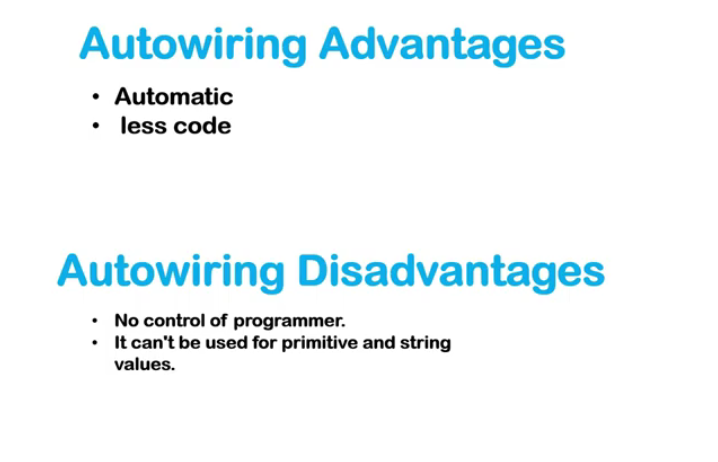
But in our case spring container will Automatic link these two classes. This is Auto-wiring.



**Auto wiring –** We can achieve auto-wiring by two ways.

1. XML auto-wiring
2. Annotation auto-wiring



****

* If we are using byName mode for autowiring then it will look for the bean which name is same as reference variable name and if we didn’t have same reference variable name then we will get null output.
* If we are using byType mode for autowiring then it will look for the bean which have same type (i.e. class name) and if we have two type with same name then we will get exception.
* If we are using constructor mode for autowiring then it will look for constructor and but it will work when the reference variable is same as bean name.

**@Autowired** :- We can use this annotation on property, setter method, constructor. When we use this annotation, it will not check the name of reference variable or bean name as it is following the concept of **byType.**

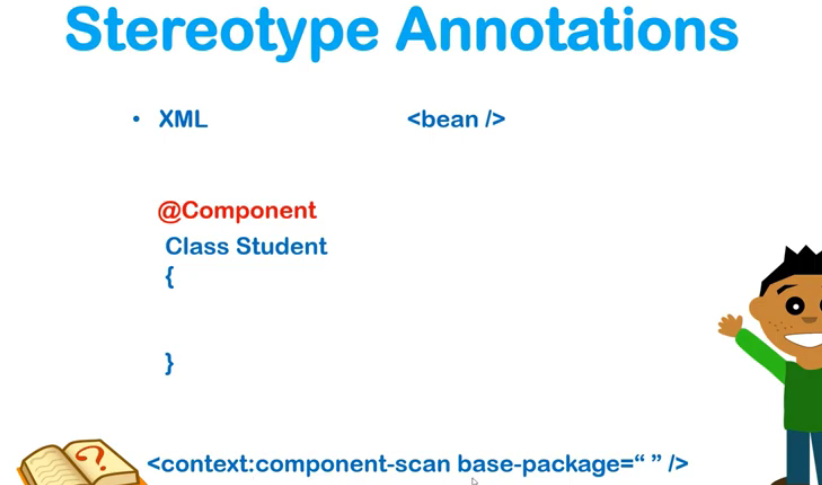
**Why we need @Qualifier annotation?**

@Qualifier annotation: - If we have more than one bean of same **Type** then we can use this annotation and mention that we want to execute a particular bean**.**

**Syntax: -** @Qalifier(“bean name”)

**Standalone Collections:** -

**Stereotype Annotations**

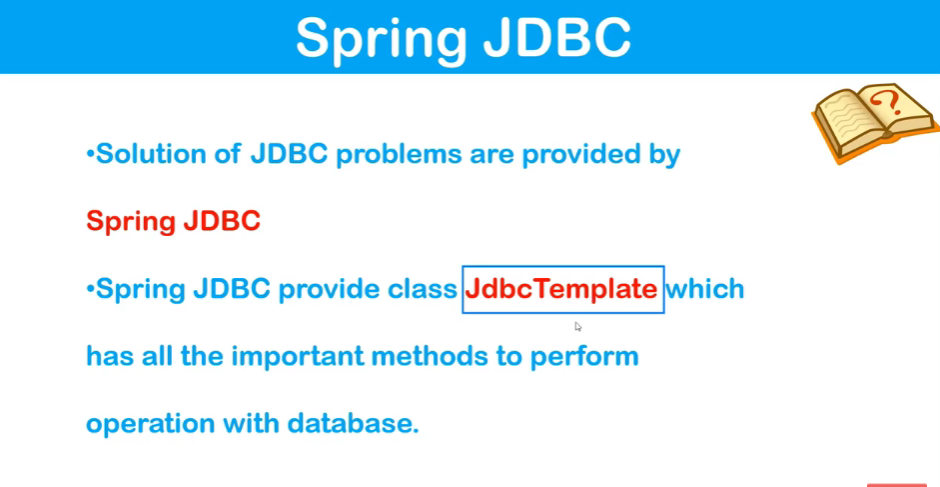
****

Till now we are using bean for object creation now we just mention @Component annotation so it will automatically create object of that class.

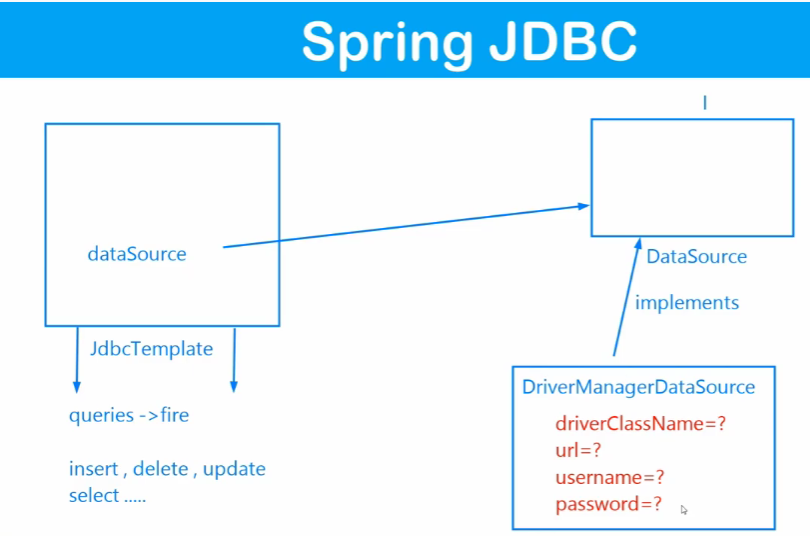
@Value annotation is used to specify the values for variable.

**Spring JDBC**

Spring jdbc is powerful mechanism to connect database and execute SQL queries.

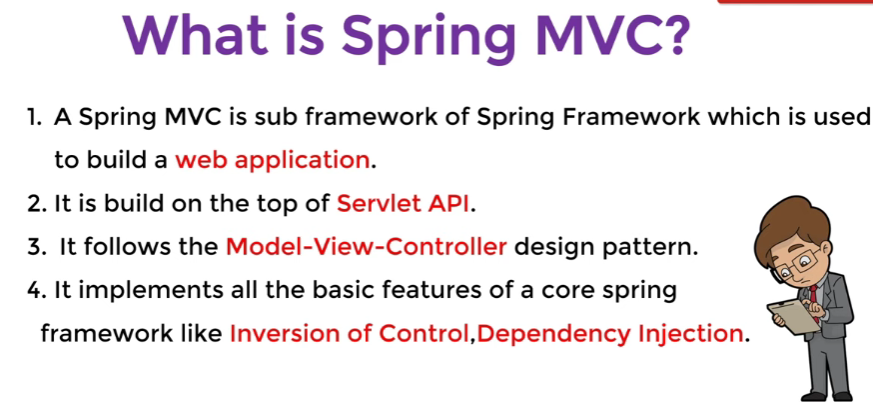


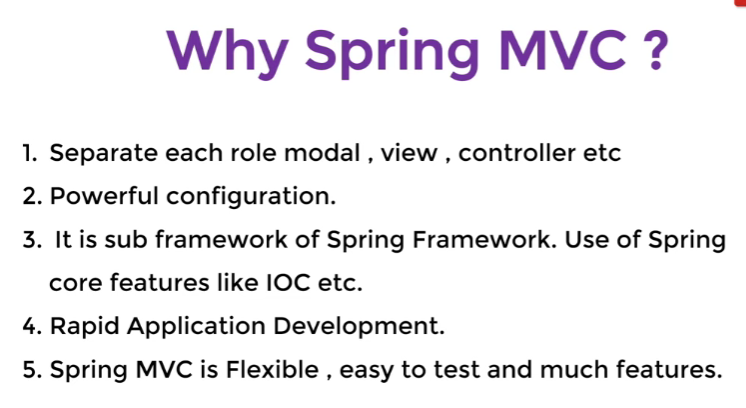
Datasource has all the information about database.

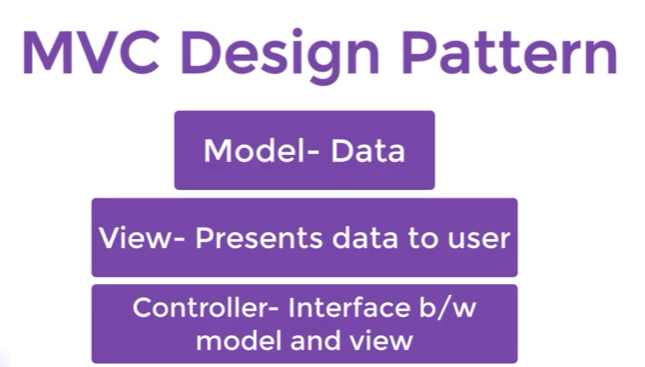


As DataSource is an interface so we cannot create object of it. So, DriverManagerDataSource implement DataSource. It will create the object of DataSource.

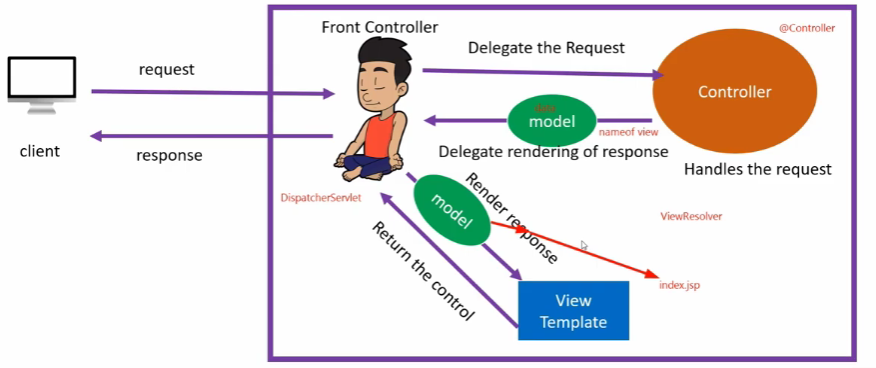
**Spring MVC**

****

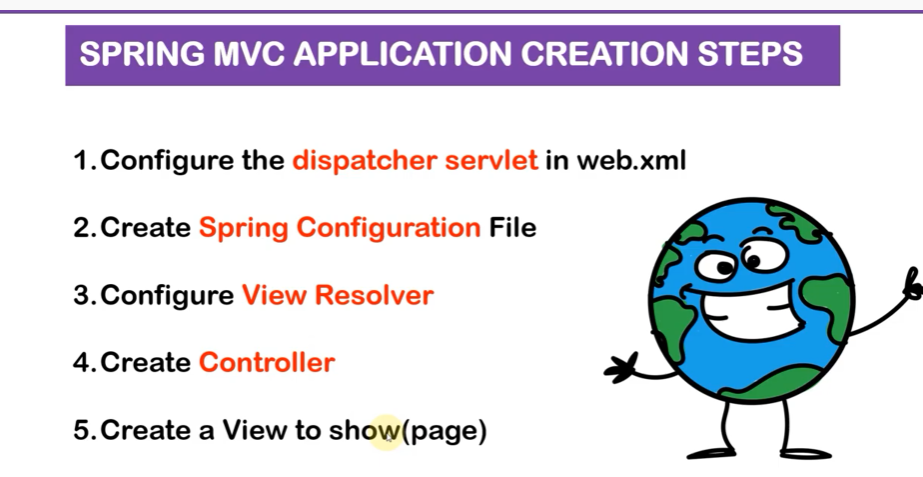
****

****

**Working Of MVC**

****

**Tomcat Server**

****

