* Annotations are started supporting from Spring 2.5V
* From Spring 4.0V its started supporting java 8 features.. spring latest version 6+.
* Spring is an application framework which is open source and loosely coupled and it reduce the complexity of Enterprise application development.
* Spring supports various frameworks like Hibernate, EJB and struts
* Spring will have POJO classes where getters and setters will be there.
* In spring there Inversion of control and dependency Injections and bean creation which made spring framework strong.
* Spring framework will has different features like light weighted, MVC, IOC, dependency Injection, Transaction Management.
* In spring framework , **Spring Inversion of control(IOC)/spring container** is the main part, it creates the bean(objects) and configure them and manages it, it uses dependency injection for this,

**Benefits of IOC**:

* It will minimize the amount of code in your application.
* It will make your application easy to test because it doesn’t require any singletons object
* It promotes loose coupling

**Dependency Injection:**

* Spring core is a part of Dependency Injection, where **dependency Injection** will avoid the problems like where one Objects injects the data to another objects which is dependent.
* This is avoided by loosely coupling between the classes can be possible by using interface where the common functionality and implementation will be different but functionality will be remains same. So what ever we need we can use directly not depending upon one. This configurations will be done by developers.
* This **Injection of dependence** is done by either Setters or Constructor’s.
* In **Transaction management** we need write lot of code here, Spring provide some inbuilt feature like **Generic Abstraction Layers**. @Transactional 🡺 update, delete

Beans:

* Beans are nothing but Objects and this is created by the IOC and it plays a crucial role.
* Beans can be created by the XML configuration and by Annotations Based configurations  
  @Bean, @Configuration : both are used annotation based.
* Bean is by default like singleton object once we create and we use it where ever it requires,
* It is not a good way to use spring beans, since because it works as I need a object but I didn’t requires from where it Is coming from.
* Beanfactory is like combined of beans.

@Component: This makes the Java class as Bean, so the component-scanning mechanism of spring can pick it up and pull it into the configuration while running application.

Ex:

@Configuration

@Component

Class Student {

@Bean(name=”…”)

Public Student student(){

return new Student();

}

In Spring, @Component , @Bean  annotations are quite different.

@Component used to auto-detect and auto-configure beans using classpath scanning.

@Bean is used to explicitly declare a single bean, rather than letting Spring do it automatically for us.

Another big difference is that @Component is a **class level annotation** where as @Bean is a **method level annotation** and ,by default, name of the method serves as the bean name.

* Stereotype Anotations:

In spring there are 4 Anotations in Stereotype i.e, @Components, @Controller @Service @Repository

@Controller:

This says that class is a Controller class and beans marked to it will be directly injected through the dependency injections. And all URLs related code will placed here.

@Service: @component

It is used for writing all the business logics and it it part of components annotation

the basic difference is that @component is generic type and object will create for any use but on the other hand @service is a type of component and its object is also created by IOC container but its purpose is to define the bussiness logic.

@repository:

It is used in Database Layer, which says that it is DAO and integrates with Service layer.

**It also makes the unchecked exceptions (thrown from DAO methods) eligible for translation** into Spring DataAccessException.

For example, Spring Data JPA will process @Repository, and will try to replace with implementation any interface marked by this annotation. Spring also will apply automatic exception translation to such classes

@Autowired annotation:

It is like without creating the objects we will inject the beans, it used at setters and constructors and Methods above.

@Qualifier:

When you create more than one bean of the same type and want to wire only one of them with a property  you can use the **@Qualifier** annotation along with **@Autowired** to remove the ambiguity by specifying which exact bean should be wired.

noSuchBeanDefinitionException will through if there is same beans.

@RequestMapping annotation is used for mapping a particular HTTP request method to a specific class/ method in controller that will be handling the respective request. This annotation can be applied at both levels:

* **Class level**: Maps the URL of the request

Controller

@autowire

serviceInterface

serviceMethod(data)

Service-- interface

Impl1

serviceMethod(data)

Impl2

serviceMethod(data)