We have learnt 2 methods to create bean object till now.

1. Using <bean> tag
2. Using @Component

But there is 3rd way as well using **@Bean** annotation.

So **@Bean** annotation is simply use for creating bean objects. How we going to create bean object using this annotation let’s look into practical,

**College.java :**

public class College {

}

This empty is going to be instantiate.

**CollegeConfig.java :**

@Configuration

public class CollegeConfig {

@Bean

public College collegeBean() {

return new College();

}

}

This is our configuration file. Inside it we have a method which is returning the College class object. As we have annotated it using **@Bean** annotation, so it is a signal for spring to call this method and whatever object returning by the method should be push inside the IOC container.

**Client.java :**

public class Client {

public static void main(String[] args) {

ApplicationContext context = new AnnotationConfigApplicationContext(CollegeConfig.class);

College stu = context.getBean("collegeBean", College.class);

System.out.println(stu);

}

}

Now the code will run successfully, but a question should be arise in your mind that we are accessing the College class object from context using “collegeBean” id

College stu = context.getBean("collegeBean", College.class);

Although, we never assign any id anywhere to the object. **So how did this happen ?**

Answer is, although we can give multiple id’s to our bean object using **name** attribute of **@Bean** annotation but if we don’t provide any id as we have done above so spring make the method name id of bean object.

**How can we give multiple id’s to our bean object ?**

@Bean( name = {“a”, “b”, “c”} )

If you want to give only single id, you can go like below :

@Bean( name = “a” )

But if you gave your id to the bean object then the default will be overwrite. You have access the object only the id you have given to **name** attribute.

**What If College Class Has Dependency Of Principle Class ?**

Take a look below project you will get a fare idea about it,

**Principal.java :**

public class Principal {

@Override

public String toString() {

return "Hi from principal";

}

}

**College.java :**

public class College {

private Principal principal;

public void setPrincipal(Principal principal) {

this.principal = principal;

}

public void helloPrincipal() {

System.out.println(principal);

}

}

**CollegeConfig.java :**

uration

public class CollegeConfig {

@Bean

public Principal principalBean() {

return new Principal();

}

@Bean

public College collegeBean() {

College clg = new College();

clg.setPrincipal(this.principalBean());

return clg;

}

}

**Client.java :**

public class Client {

public static void main(String[] args) {

ApplicationContext context = new AnnotationConfigApplicationContext(CollegeConfig.class);

College stu = context.getBean("collegeBean", College.class);

stu.helloPrincipal();

}

}