

## 02 - Using Annotations in SpringBoot

Spring Boot supports all the annotations from Spring, simplifying configurations and reducing the boilerplate code. These annotations help in managing dependencies, injecting values, and defining beans.

### 👉 Key Annotations used to simplify the configuration:

**@Component:** The @Component annotation marks a Java class as a Spring-managed component, allowing Spring to auto-detect and register the bean through classpath scanning.

**@Value:** The @Value annotation is used to inject values directly from properties files or environment variables into Spring-managed beans. It reduces the need for manual configuration in XML or Java code.

**@Autowired:** The @Autowired annotation is used for dependency injection in Spring. It tells Spring to automatically inject the required beans into the class's fields, constructor, or methods.

**@Qualifier:** The @Qualifier annotation helps in disambiguating which bean should be injected when there are multiple beans of the same type.

**@Primary:** The @Primary annotation is used to specify which bean should be given preference when multiple beans of the same type exist. If no @Qualifier is provided, Spring will inject the bean annotated with @Primary.

**@Bean:** It indicates that a method will return a Spring bean to be managed by the Spring container.

## Example:

### *Alien.java*

```
@Component
public class Alien {

    @Value("25")
    private int age;
    private Computer comp;

    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
    public Computer getComp() {
        return comp;
    }

    @Autowired
    @Qualifier("laptop")
    public void setComp(Computer comp) {
        this.comp = comp;
    }

    public void code() {
        comp.compile();
    }
}
```

### *Computer.java*

```
public interface Computer {
    void compile();
}
```

### ***Desktop.java***

```
@Component
@Primary
public class Desktop implements Computer{
    public void compile() {
        System.out.println("Compiling in Desktop");
    }
}
```

### ***Laptop.java***

```
@Component
public class Laptop implements Computer{

    public void compile() {
        System.out.println("Compiling in Laptop");
    }
}
```

### ***SpringBootDemoApplication.java***

```
@SpringBootApplication
public class SpringBootDemoApplication {

    public static void main(String[] args) {
        ApplicationContext context=
            SpringApplication.run(SpringBootDemoApplication.class, args);

        Alien obj=context.getBean(Alien.class);
        System.out.println(obj.getAge());
        obj.code();
    }
}
```

### **Code Link:**

[https://github.com/navinreddy20/spring6-course/tree/c6690e4f2c70d8f530d70623f13d14ff0ffd7e7d/3.2%20Moving%20to%20Spring%20Boot%20\(3.10%20to%203.14\)/3.11%20Using%20Annotations%20In%20Spring%20Boot](https://github.com/navinreddy20/spring6-course/tree/c6690e4f2c70d8f530d70623f13d14ff0ffd7e7d/3.2%20Moving%20to%20Spring%20Boot%20(3.10%20to%203.14)/3.11%20Using%20Annotations%20In%20Spring%20Boot)