Session-27

Spring - Annotation Based Configuration

Annotations:

Java **Annotation** is a tag that represents the *metadata* i.e. attached with class, interface, methods or fields to indicate some additional information which can be used by java compiler and JVM.

Annotations in Java are used to provide additional information, so it is an alternative option for XML and Java marker interfaces.

Spring Annotations:

Starting from Spring 2.5 it became possible to configure the dependency injection using **annotations**. So instead of using XML to describe a bean wiring, you can move the bean configuration into the component class itself by using annotations on the relevant class, method, or field declaration.

- ✓ Spring framework implements and promotes the principle of control inversion (IOC) or dependency injection (DI) and is in fact an IOC container.
- ✓ Traditionally, Spring allows a developer to manage bean dependencies by using XML-based configuration.
- ✓ There is an alternative way to define beans and their dependencies. This method is a Java-based configuration.
- ✓ Unlike the XML approach, Java-based configuration allows you to manage bean components programmatically. That's why Spring annotations were introduced.

Annotation injection is performed before XML injection. Thus, the latter configuration will override the former for properties wired through both approaches.

Annotation wiring is not turned on in the Spring container by default. So, before we can use annotation-based wiring, we will need to enable it in our Spring configuration file.

So consider the following configuration file in case you want to use any annotation in your Spring application.

```
<?xml version = "1.0" encoding = "UTF-8"?>

<beans xmlns = "http://www.springframework.org/schema/beans"
  xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
  xmlns:context = "http://www.springframework.org/schema/context"
  xsi:schemaLocation = "http://www.springframework.org/schema/beans
  http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
  http://www.springframework.org/schema/context
  http://www.springframework.org/schema/context/spring-context-3.0.xsd">

  <context:annotation-config/>
  <!-- bean definitions go here -->

</beans>
```

Once <context:annotation-config/> is configured, you can start annotating your code to indicate that Spring should automatically wire values into properties, methods, and constructors. Let us look at a few important annotations to understand how they work

Spring Annotations List:

Some of the spring core framework annotations are:

1. @Configuration: Used to indicate that a class declares one or more @Bean methods. These classes are processed by the Spring container to generate bean definitions and service requests for those beans at runtime.

2. @Bean: Indicates that a method produces a bean to be managed by the Spring container. This is one of the most used and important spring annotation. @Bean annotation also can be used with parameters like name, initMethod and destroyMethod.

- name – allows you give name for bean
- initMethod – allows you to choose method which will be invoked on context register
- destroyMethod – allows you to choose method which will be invoked on context shutdown

For example:

```
@Configuration
public class AppConfig {

  @Bean(name = "comp", initMethod = "turnOn", destroyMethod = "turnOff")
  Computer computer(){
    return new Computer();
  }
}

public class Computer {

  public void turnOn(){
    System.out.println("Load operating system");
  }
  public void turnOff(){
    System.out.println("Close all programs");
  }
}
```

3. @PreDestroy and @PostConstruct are alternative way for bean initMethod and destroyMethod. It can be used when the bean class is defined by us. For example;

   ```
   public class Computer {

     @PostConstruct
     public void turnOn(){
       System.out.println("Load operating system");
     }

     @PreDestroy
     public void turnOff(){
       System.out.println("Close all programs");
     }
   }
   ```

4. @ComponentScan: Configures component scanning directives for use with @Configuration classes. Here we can specify the base packages to scan for spring components.

5. @Component: Indicates that an annotated class is a “component”. Such classes are considered as candidates for auto-detection when using annotation-based configuration and classpath scanning.

6. @PropertySource: provides a simple declarative mechanism for adding a property source to Spring’s Environment. There is a similar annotation for adding an array of property source files i.e @PropertySources.

7. @Service: Indicates that an annotated class is a “Service”. This annotation serves as a specialization of @Component, allowing for implementation classes to be autodetected through classpath scanning.

8. @Repository: Indicates that an annotated class is a “Repository”. This annotation serves as a specialization of @Component and advisable to use with DAO classes.

9. @Autowired: Spring @Autowired annotation is used for automatic injection of beans. Spring @Qualifier annotation is used in conjunction with Autowired to avoid confusion when we have two of more bean configured for same type.