**7. Spring Configuration with Java Annotations - Inversion of Control**

**Java Annotation**:

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.

In a word java annotation is simple metadata about a class. Annotation provide important message to the compiler or JVM.

**Why spring configuration with annotation**:

public class TrackCoach implements Coach {  
 @Override  
 public String getDailyWorkout() {  
 return "Run a hard 5k";  
 }  
 …  
}

**Why spring configuration with annotation**:

1. XML configuration can be verbose
2. Configure your beans with annotations
3. Annotations minimize the XML configuration.

**Scanning for Component Classes**:

Basically once we add an annotation to a class, then spring will actually scan the Java classes for those annotations. When it finds a class that has a special annotation on it, it’ll automatically register that bean with the Spring container

**Development process**:

1. Enable component scanning in Spring configuration file
2. Add the @Component Annotation to your Java classes
3. Retrieve bean from Spring container

**1) Enable component scanning in Spring configuration file**:

<beans … >

<context:component-scan base-package="com.odduu.ruhul" />

</beans>

**2) Add the @Component Annotation to your Java classes**:

@Component("thatSillyCoach")

public class TennisCoach implements Coach {

@Override

public String getDailyWorkout() {

return "Practice your backhand volley";

}

}

**3) Retrieve bean from Spring container**:

Coach theCoach = context.getBean("thatSillyCoach", Coach.class);

**Example**:

**File: Coach.java**:

**package** com.ruhul.odduu;

**public** **interface** Coach {

**public** String getDailyWorout();

}

**File: TennisCoach.java**:

**package** com.ruhul.odduu;

**import** org.springframework.stereotype.Component;

@Component("thatSillyCoach")

**public** **class** TennisCoach **implements** Coach {

@Override

**public** String getDailyWorout() {

**return** "Practice your backhand volley";

}

}

**File: applicationContext.xml**:

<?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.xsd*

*http://www.springframework.org/schema/context*

*http://www.springframework.org/schema/context/spring-context.xsd"*>

<!-- add entry to enable component scanning -->

<context:component-scan base-package=*"com.ruhul.odduu"* />

</beans>

**File: AnnotationDemoApp.java**:

**package** com.ruhul.odduu;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**public** **class** AnnotationDemoApp {

**public** **static** **void** main(String[] args) {

// read spring congig file

ClassPathXmlApplicationContext context = **new** ClassPathXmlApplicationContext("applicationContext.xml");

// get the bean from container

Coach theCoach = context.getBean("thatSillyCoach", Coach.**class**);

// call a method on the bean

System.***out***.println(theCoach.getDailyWorout());

// close the context

context.close();

}

}

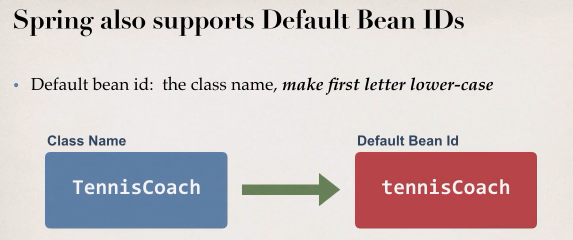
**Output**:

Practice your backhand volley

**@Components-Default Bean ID**:

When Spring creates a Bean Definition from a @Service or @Component annotation, it will by default create an id for the bean by lowercasing the first letter of the Class Name.

We can also provide different name.



**Example**:

**File: TennisCoach.java**:

**package** com.ruhul.odduu;

**import** org.springframework.stereotype.Component;

@Component

**public** **class** TennisCoach **implements** Coach {

@Override

**public** String getDailyWorout() {

**return** "Practice your backhand volley";

}

}

**File: AnnotationDemoApp.java**:

**package** com.ruhul.odduu;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**public** **class** AnnotationDemoApp {

**public** **static** **void** main(String[] args) {

// read spring congig file

ClassPathXmlApplicationContext context = **new** ClassPathXmlApplicationContext("applicationContext.xml");

// get the bean from container

Coach theCoach = context.getBean("tennisCoach", Coach.**class**);

// call a method on the bean

System.***out***.println(theCoach.getDailyWorout());

// close the context

context.close();

}

}

/\*

Output:

Practice your backhand volley

\*/

7. Spring Configuration with Java Annotations - Inversion of Control