Spring Configuration with Annotation

Wikipedia definition of Annotation:
In the Java computer programming language, an **annotation** is a form of syntactic metadata that can be added to Java source code.Classes, methods, variables, parameters and packages may be annotated.

Annotation example:
@Override => Tell the compiler we are overriding method. Generally run at compile time.

Why Annotation:
1. To minimize the xml config.
2. Config your spring beans with annotation

Example: spring-annotations-module-4

Development Process steps:
1. Enable component scanning in the spring config file.

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

     <context:component-scan base-package="com.branch.springwork"></
context:component-scan>
```

2. Add the @Component annotation to your java classes

```
@Component("eng") //eng is the bean id
public class EngineeringTeam implements Manager {
 . . . .
 . . . .
}
```

3. Retrieve bean from spring container

```
//retrieve bean from spring container
Manager manager = context.getBean("eng", Manager.class);
```

Notes:

1. If you compare the code in applicationContext.xml with previous example in module-2 then you notice that the lines of code is dramatically reduces. This is because you are using annotation.

2. In above step-2, we have written @Component(“**eng**”). eng is the bean id by which we retrieve the bean from spring container.

```
Manager manager = context.getBean("eng", Manager.class);
```

3. If you don’t specify the bean id in step-2 then spring uses default bean id which is class name. However, the first letter of the class name is lower case.

For example:

```
@Component
public class EngineeringTeam implements Manager {
 . . . .
 . . . .
}

//retrieve bean from spring container
Manager manager = context.getBean("engineeringTeam", Manager.class);
```

Notice that class name is: **EngineeringTeam**

And corresponding bean id is: **engineeringTeam**