Spring Boot – Implementing Lombok

Description

Whenever we create (POJO's) plain old Java objects in our application, we subconsciously create getter, setters, toString() method and sometimes constructors as well. But have you ever thought that by doing this your class becomes too lengthy if the attributes are more. There is one way where you can remove all the setter getter methods and just keep the class attributes. Lombok provides annotation which you can use based on your project requirements. Lets see what all things are required to setup Lombok in your application before exploring all the annotations.

1. Adding the dependency

You need to add the below maven dependency in your pom.xml. If you want to know more about Lombok or the latest version, then you can head over to lombok website

default watermark

```
<dependencies>
    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <version>1.18.20</version>
    </dependency>
</dependencies>
```

2. Replacing Getters and Setters using lombok's @Getter and @Setter

You will have to annotate your Domain objects with @Getter & @Setter annotation. Lombok will automatically generate the default getter/setter for you.

```
@Getter
@Setter
public class Employee {
 private String empId;
 private String empName;
 private Address address;
 .
 .
}
```

In the above code snippet, Annotations are added at the class level. You can also add it on the field level based on your requirements. The generated getter/setter method will be `public` unless you explicitly specify an `AccessLevel`, as shown in the example below.

Legal access levels areÂ `PUBLIC`,Â `PROTECTED`,Â `PACKAGE`, andÂ `PRIVATE`.

```
public class Employee {
 @Getter
 @Setter
 private String empId;
 @Setter(AccessLevel.PROTECTED)
 private String empName;
 private Address address;
 .
 .
}
```

If you want to know more in-depth about the two annotations, then head over to this link

3. @ToString andÂ @EqualsAndHashCode

`@ToString` and `@EqualsAndHashCode` will be used to generate the toString, equals and hashcode methods. You can add these annotation above your class.

default watermark

4. Adding Constructors using @NoArgsConstructor, @RequiredArgsConstructor, @AllArgsConstructor

`@NoArgsConstructor` annotation will generate a constructor which will not any parameters. If this is not possible (because of final fields), a compiler error will result instead, unless **`@NoArgsConstructor(force = true)`** is used, then all final fields are initialized with `0` / `false` / `null`

`@RequiredArgsConstructor` generates a constructor with 1 parameter for each field that requires special handling. All non-initialized `final` fields get a parameter, as well as any fields that are marked as `@NonNull` that aren't initialized where they are declared. For those fields marked with `@NonNull`, an explicit null check is also generated. The constructor will throw a `NullPointerException` if any of the parameters intended for the fields marked with `@NonNull` contain `null`. The order of the parameters match the order in which the fields appear in your class.

`@AllArgsConstructor`Â generates a constructor with 1 parameter for each field in your class. Fields marked withÂ `@NonNull`Â result in null checks on those parameters

```
@NoArgsConstructor
@AllArgsConstructor
public class Employee {
 @Getter
 @Setter
 private String empId;
 @Setter(AccessLevel.PROTECTED)
 private String empName;
```

```
 private Address address;
 .
 .
}
```

4. @Data annotation

@Data annotation is a replacement for `@ToString`, `@EqualsAndHashCode`, `@Getter` on all fields, `@Setter` on all non-final fields, and `@RequiredArgsConstructor`.

@Data = `@ToString` +Â `@EqualsAndHashCode` +Â `@Getter` + `@Setter` + @RequiredArgsConstructor

```
@Data
public class Employee {
 @Getter
 @Setter
 private String empId;
 @Setter(AccessLevel.PROTECTED)
 private String empName;
 private Address address;
 .
 .
}
```

default watermark

After you have added all your required annotations, If you want to see the methods that got generated you can open the .class files after you build your application.

I hope you liked this post. If you have any questions or queries feel free to comments or write to us.

Category

1. Hands on
2. Lombok

Date Created
July 11, 2021
Author
kk-ravi144gmail-com