Day16: Lombok and Swagger

Lombok Java

Java is the most popular object-oriented programming language but it has some drawbacks. The major drawback is to write lots of **boilerplate** code. To overcome this drawback, project **Lombok** comes into existence. It is a tool that spices up our Java application. In this section, we will discuss the **project Lombok, features, Lombok package** .

What is project Lombok?

The project Lombok is a popular and widely used Java library that is used to minimize or remove the boilerplate code. It saves time and effort. Just by using the annotations, we can save space and readability of the source code. It is automatically plugging into IDEs and build tools to spice up our Java application.

Here, a question arises that **does project Lombok and IDEs do the same work? If yes, then what is the use of Lombok?**

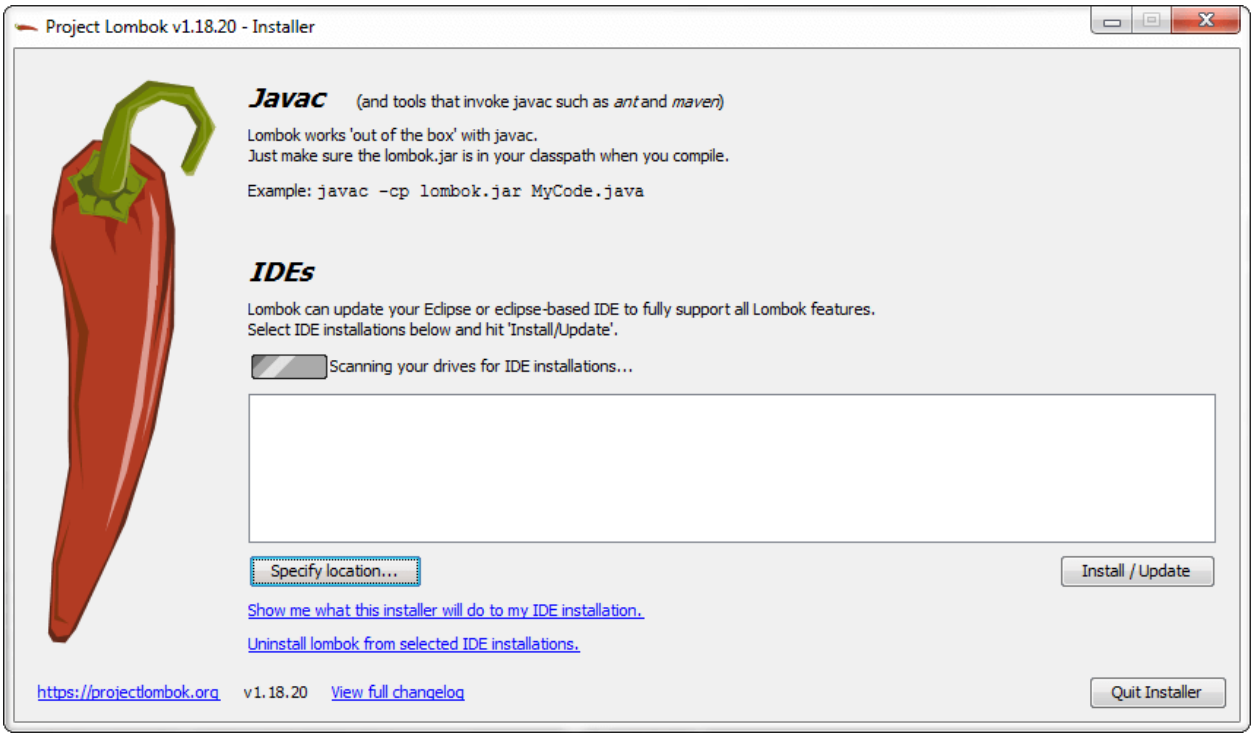
The answer is **no**, IDEs and Lombok do different works but are closely similar to each other. When we use IDEs to generate these boilerplate codes (getters and setters), we save ourselves from writing getters and setters manually but it actually exists in the source code that increases the lines of code, and reduces maintainability and readability. While the project Lombok adds all these boilerplate codes at the compile-time in the class file instead of adding these boilerplate code in original source code.

Configure Lombok in STS:

To configure the Lombok project in STS IDE, follow the steps given below:

Step 1: First, download the lombok.jar file.

Step 2: For executing the above JAR file, double-click on the downloaded JAR file. A GUI appears on the screen in which we have to specify the IDE on which we want to configure the Lombok project.



Step 3:

Click on the **Specify location**

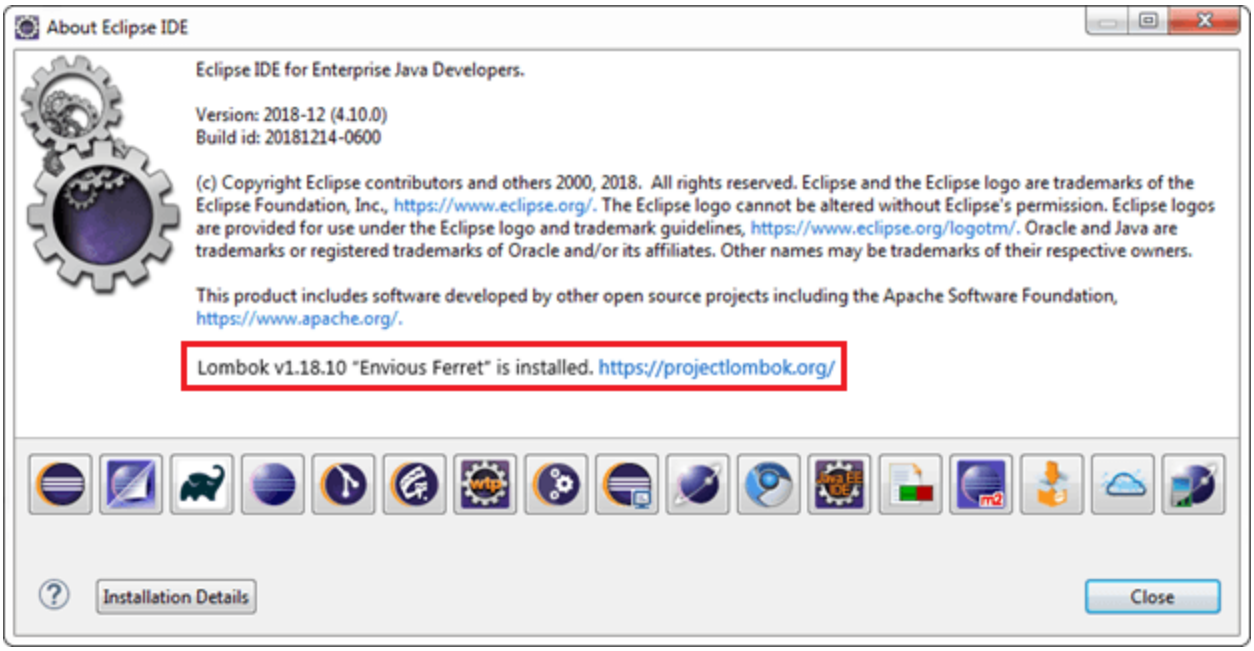
button and browse the directory in which STS IDE is installed. From the folder select springtoolsuit**.exe** file.

Step 4: Click on the **Install/ Update** button.

Once the above process is completed check project Lombok is installed successfully or not.

Step 5: Open STS IDE -> **Help** -> **About STS IDE**

. If the project Lombok is listed, means that properly installed.



Step 6: At last, click on the **Close** button.

The project Lombok is successfully integrated with the STS IDE.

Java Lombok provides various annotations to avoid the boiler plate code:

Annotations	Description
AllArgsConstructor	Generates an all-args constructor.
Data	Generates getters for all fields, a useful toString method, and hashCode and equals implementations that check all non-transient fields.
EqualsAndHashCode	Generates implementations for the equals and hashCode methods inherited by all objects, based on relevant fields.
EqualsAndHashCode.Exclude	If present, do not include this field in the generated equals and hashCode methods.
EqualsAndHashCode.Include	Configure the behavior of how this member is treated in the equals and hashCode implementation; if on a method, include the method's return value as part of calculating hashCode/equality.
Getter	Put on any field to make Lombok build a standard getter.

NoArgsConstructor	Generates a no-args constructor.
NonNull	If put on a parameter, Lombok will insert a null-check at the start of the method /constructor's body, throwing a NullPointerException with the parameter's name as a message.
RequiredArgsConstructor	Generates a constructor with required arguments.
Setter	Put on any field to make Lombok build a standard setter.
SneakyThrows	@SneakyThrow will avoid javac's insistence that you either catch or throw onward any checked exceptions that statements in your method body declare they generate.
Synchronized	Almost exactly like putting the 'synchronized' keyword on a method, except will synchronize on a private internal Object, so that other code not under your control doesn't meddle with your thread management by locking on your own instance.
ToString	Generates an implementation for the toString method inherited by all objects, consisting of printing the values of relevant fields.
ToString.Exclude	If present, do not include this field in the generated toString.
ToString.Include	Configure the behavior of how this member is rendered in the toString; if on a method, include the method's return value in the output.

Swagger2:

Nowadays, front-end and back-end components often separate a web application. Usually, we expose APIs as a back-end component for the front-end component or third-party app integrations.

In such a scenario, it is essential to have proper specifications for the back-end APIs. At the same time, the API documentation should be informative, readable, and easy to follow.

Moreover, reference documentation should simultaneously describe every change in the API. Accomplishing this manually is a tedious exercise, so automation of the process was inevitable.

In this tutorial, we'll look at **Swagger 2 for a Spring REST web service**, using the Springfox implementation of the Swagger 2 specification. If you are not familiar with Swagger,

Adding the Maven Dependency

For the Spring Boot based projects, **it's enough to add a single *springfox-boot-starter* dependency**
:

```
<dependency>
    <groupId>io.springfox</groupId>
    <artifactId>springfox-boot-starter</artifactId>
    <version>3.0.0</version>
</dependency>
```