Introduction to Project Lombok in Java and How to get started?

Difficulty Level : Easy • Last Updated : 20 Jun, 2022

Java is a very popular language, but it has a few drawbacks. One of the most popular drawbacks is that we still need to write the boilerplate codes like getters, setters, and toString() method in Java whereas Kotlin and Scala, which are also JVM based don't need so, and hence, this is the reason for their increased popularity in the community. This is where Lombok comes into the picture and overcomes this drawback of Java. Project Lombok is a java library tool that is used to minimize/remove the boilerplate code and save the precious time of developers during development by just using some annotations. In addition to it, it also increases the readability of the source code and saves space. But you might be thinking that nowadays, everyone uses IDEs which provides an option for generating these boilerplate codes, then what is the use of Lombok. Whenever we use IDEs to generate these boilerplate codes, we just save ourselves from writing all these codes but it is actually present in our source code and increases the LOC (lines of code), and reduces maintainability and readability. On the other hand, Lombok adds all these boilerplate codes at the compile time in the ".class" file and not in our source code. Let us compare our source code with and without using Lombok.

1. Without Lombok: A java model class with four private fields and their getters, setters, no-args constructor, parameterized construct, and toString method.

Java

```
public class Employee {
```

```
    public Employee(Integer employeeId, String name,
                    String company, String emailId)
    {
        super();
        this.employeeId = employeeId;
        this.name = name;
        this.company = company;
        this.emailId = emailId;
    }

    public Integer getEmployeeId() { return employeeId; }

    public void setEmployeeId(Integer employeeId)
    {
        this.employeeId = employeeId;
    }

    public String getName() { return name; }

    public void setName(String name) { this.name = name; }

    public String getCompany() { return company; }

    public void setCompany(String company)
    {
        this.company = company;
    }

    public String getEmailId() { return emailId; }

    public void setEmailId(String emailId)
    {
        this.emailId = emailId;
    }

    @Override public String toString()
    {
        return "Employee ["
            + "employeeId=" + employeeId + ", name=" + name
            + ", "
            + " company=" + company + ", emailId=" + emailId
            + "]";
    }
}
```

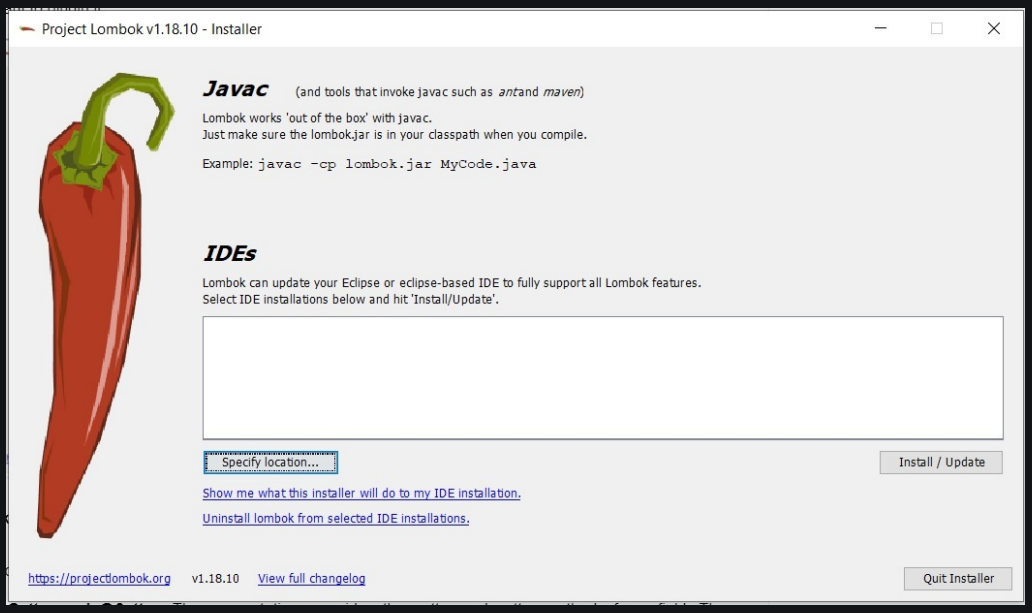
Java

```
import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import lombok.ToString;

@NoArgsConstructor
@AllArgsConstructor
@ToString
public class Employee {
    private @Getter @Setter Integer employeeId;
    private @Getter @Setter String name;
    private @Getter @Setter String company;
    private @Getter @Setter String emailId;
}
```

The difference is clearly visible in the codes. The original source code contains around 60 lines of code which has been reduced by Lombok to around 20 lines of code only. It not only reduces the lines of code but also increases the readability and maintainability of source code.

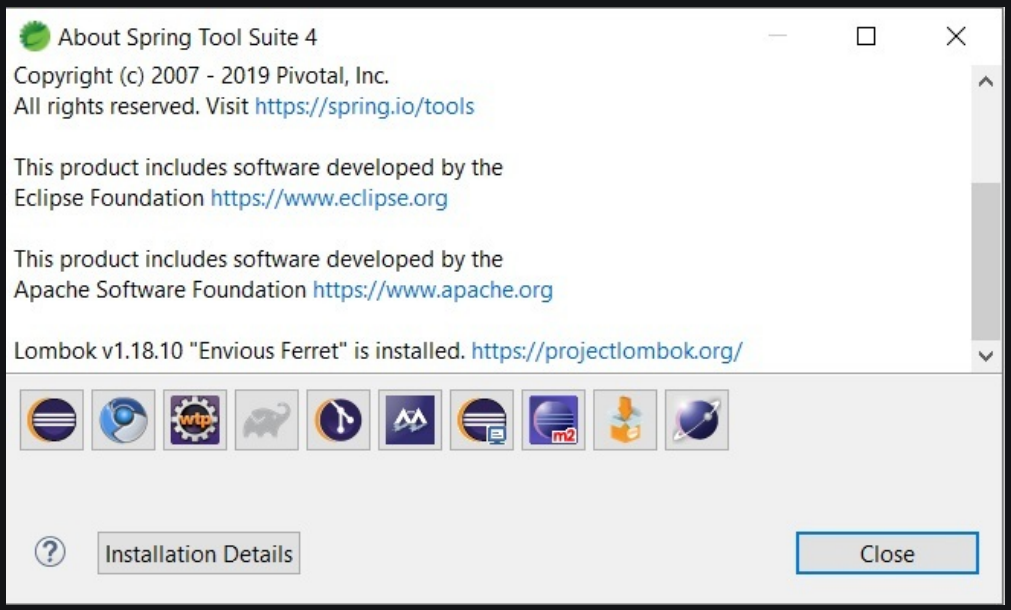
How to configure project Lombok to eclipse?



3. Click on specify a location and select the IDE.

4. Click on install to complete the installation and then quit the installer.

5. To verify the correct installation, go to the Eclipse IDE, select help, and then about the eclipse. You will see project Lombok installed there.



1. @Getter and @Setter: These annotations provide the getter and setter methods for a field. These annotations can be used at both the levels, field as well as class. *@Getter* annotation generates a getter method with access type as public which simply returns the field and with name getName() if the field name is "Name". *@Setter* annotation generates a setter method with access type as public which returns void and takes a single parameter to assign the value to the field. The default setter will have the name setName() if the field name is "Name".

Java

```
import lombok.Getter;
import lombok.Setter;

public class Employee {
    private @Getter @Setter Integer employeeId;
    private @Getter @Setter String name;
    private @Getter @Setter String company;
    private @Getter @Setter String emailId;
}
```

2. @NoArgsConstructor: This annotation is used to generate a constructor with no arguments. It has an empty body and does nothing. It is generally used in combination with some other parameterized constructor in use. It is required when you want to generate an object of the class by passing no arguments in the constructor.

Java

```
import lombok.NoArgsConstructor;

@NoArgsConstructor
public class Employee {
    private Integer employeeId;
    private String name;
```

it. It is required when you want to generate an object of the class by passing the initial values of the fields in the constructor.

Java

```
import lombok.AllArgsConstructor;

@AllArgsConstructor
public class Employee {
    private Integer employeeId;
    private String name;
    private String company;
    private String emailId;
}
```

4. @ToString: This annotation is used to override the toString() method and generate a default implementation for it. The default implementation prints the class name and the fields in order, separated by commas. You can also skip some fields that you don't want to print by annotating them with *@ToString.Exclude*.

Java

```
import lombok.ToString;

@ToString
public class Employee {
    private Integer employeeId;
    private String name;
    private String company;
    private String emailId;
}
```

5. @EqualsAndHashCode: This annotation is used to override the equals() and hashCode() methods and provides a default implementation for this. The default

Java

```java
import lombok.EqualsAndHashCode;

@EqualsAndHashCode
public class Employee {
    private Integer employeeId;
    private String name;
    private String company;
    private String emailId;
}
```

6. @Data: This annotation is a shortcut annotation and bundles *@ToString*, *@Getter*, *@Setter*, *@EqualsAndHashCode* and *@RequiredArgsConstructor* annotations into a single annotation. This annotation provides all the normally used boilerplate code in the model classes of java like getters for all the fields, setter for all the non-final fields, a default implementation for toString(), equals() and hashCode() using all the fields of the class and a constructor that initializes all the fields of the class.

Java

```java
import lombok.Data;

@Data
public class Employee {
    private Integer employeeId;
    private String name;
    private String company;
    private String emailId;
}
```

7. @Builder: This annotation can be used remove boilerplate code involved in setting properties for an object. This is useful for POJO's which have many fields.With this

```
import lombok.Builder;

@Builder
public class Employee {
    private Integer employeeId;
    private String name;
    private String company;
    private String emailId;
}
```

Example: Using **@Builder**

Java

```
import java.io.*;

class LombokTest {
    public static void main (String[] args) {
        Employee employee = Employee.builder().employeeId(21).name("GFG").company
          .emailId("gfg@geeks.com").build();
    }
}
```

RECOMMENDED ARTICLES Page : 1 2 3

01 Project Idea | (Project Approval System)
13, Nov 15

02 Apache POI | Getting Started
20, Sep 18

03 Getting Started with Cross-Platform Mobile Application using Flutter
20, Nov 18

04 Project Idea | Get Your Logo
30, Jul 17

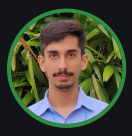
05 Project Idea | Get Me Through
08, Aug 17

06 Project Idea | Get Resources
23, Jun 18

07 Get Credential Information From the URL(GET Method) in Java
25, Aug 20

08 Project Idea | Driver distraction and drowsiness detection system - DCube
25, Jun 18

Article Contributed By :



raman_257
@raman_257

Vote for difficulty

Current difficulty : Easy

Easy | Normal | Medium | Hard | Expert

Improved By : chbs17cs, Anshul_Aggarwal, venkatanirudh36, vinamrayadav

A-143, 9th Floor, Sovereign Corporate Tower,
Sector-136, Noida, Uttar Pradesh - 201305

feedback@geeksforgeeks.org