**Lombok Introduction and Eclipse Installation**

Lombok is very handy tool for **minimizing the boilerplate code** as well as providing lot’s of other features such as **lazy loading**, **thread safety** or **immutability**. This is the reason it becoming very popular among the developer community.

In this **lombok tutorial**, we will learn about project Lombok in detail including its usage with examples.

Table of Contents

1. What is Lombok

2. Install Lombok in Eclipse

3. Using Lombok in Application

4. Delomboking - Rollback Lombok from Project

5. Summary

**1. What is Project Lombok**

Lombok is a open source library (basically a standalone jar) which is capable of doing magic in automating the boilerplate code generation for any java class. So if Lombok is in classpath, it can easily get rid of all the getters & setters methods, class constructors, hashcode and equals methods and many more by just adding couple of annotations the class.

Is it not a cool feature? Let’s start learning it to use it extensively in your next project.

**2. Install Lombok in Eclipse**

Though Lombok will work if we put Lombok on the project classpath. But in order to make it work with eclipse, we need to first do couple of steps to install it in eclipse.

1. **Download Lombok Jar File**

First we need to download the Lombok jar. We can directly download it from <https://projectlombok.org/downloads/lombok.jar> but as we will use maven in future, so let’s maven do the download on our behalf, otherwise we will have multiple version of same jar in the machine which will create problem when we will have to use updated version of it.

To do this, we will create a maven project in eclipse and add lombok dependency in pom.xml. Latest version of Lombok at the time of writing this article is **1.18.12**.

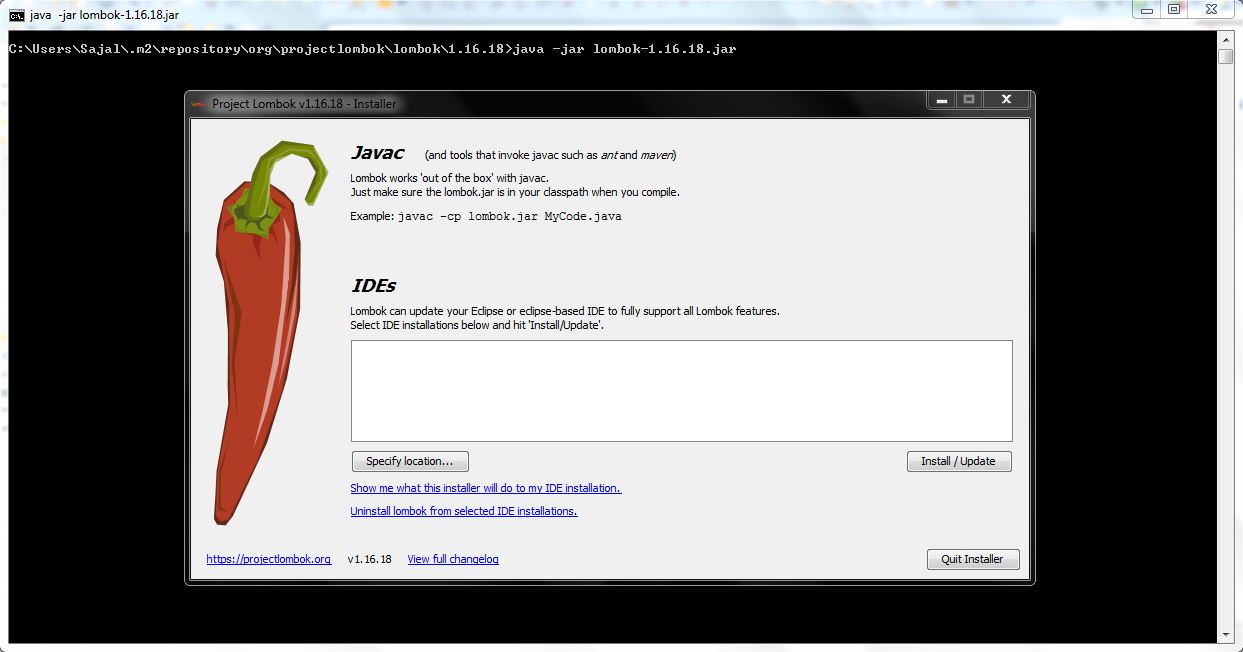
Add below dependency in your maven project so that it got downloaded first in your local repository.

|  |
| --- |
| <dependency>      <groupId>org.projectlombok</groupId>      <artifactId>lombok</artifactId>      <version>1.18.12</version>          <scope>provided</scope>  </dependency> |

Now do a mvn clean install command on the newly created project to get this jar downloaded in local repository.

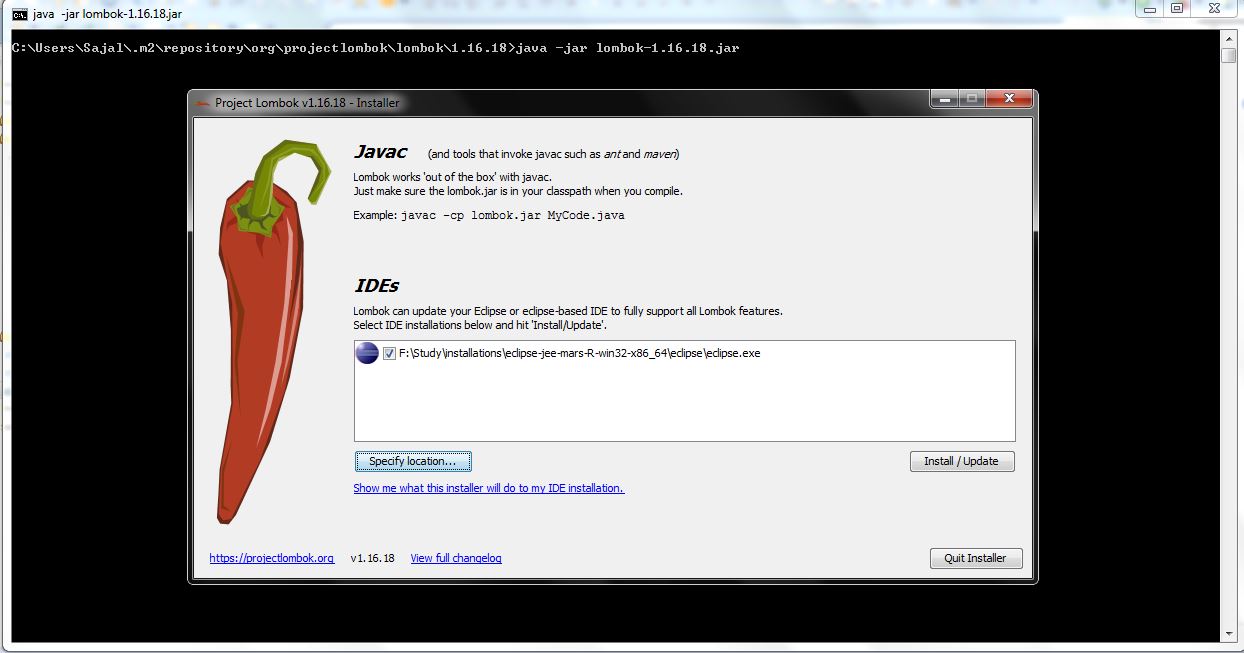
1. **Start Lombok Installation**

Once the jar downloaded in Local repository, goto the jar location from the command prompt and run the following command java -jar lombok-1.16.18.jar and we should be greeted by Lombok installation window provided by Lombok like this.

Lombok Installation in eclipse – step 1

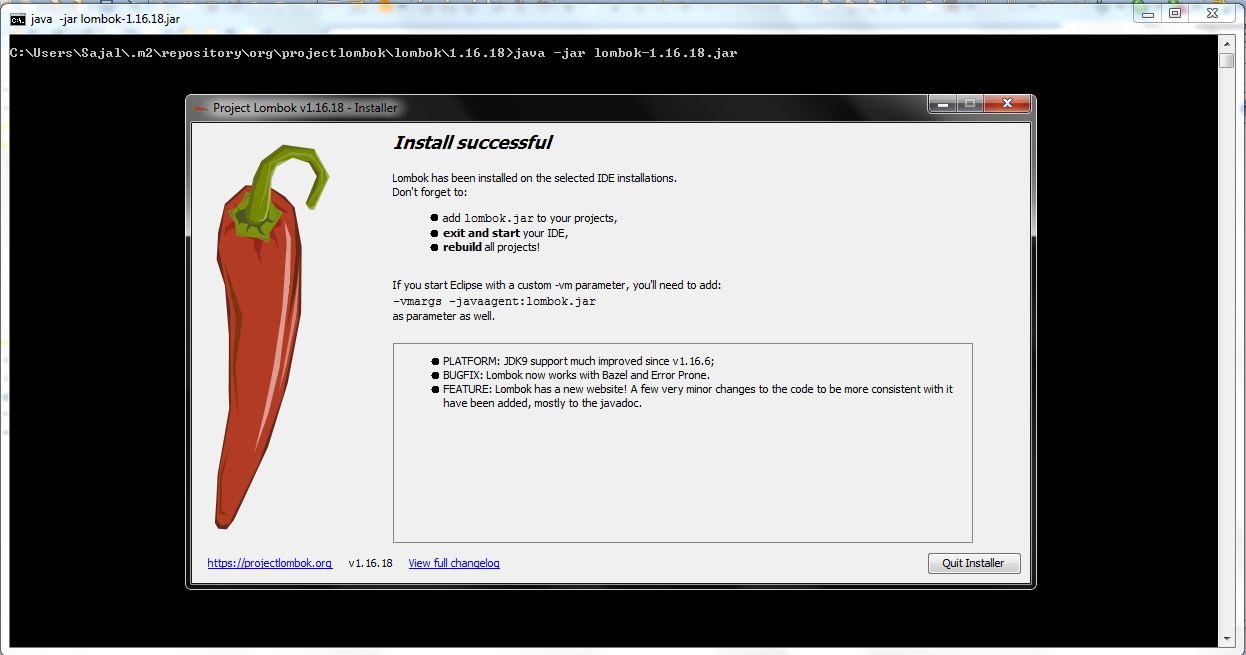
1. **Give Lombok Install Path**

Now click on the “**Specify Location**” button and locate the eclipse.exe path under eclipse installation folder like this.

Lombok Installation in eclipse – step 2

1. **Finish Lombok Installation**

Now we need to finally install this by clicking the “Install/Update” button and we should finished installing lombok in eclipse and we are ready to use its hidden power. Final screen will look like,

Lombok Installation in eclipse – step 3

**3. Using Lombok in Application**

Now let’s see some examples of using Lombok in project source code.

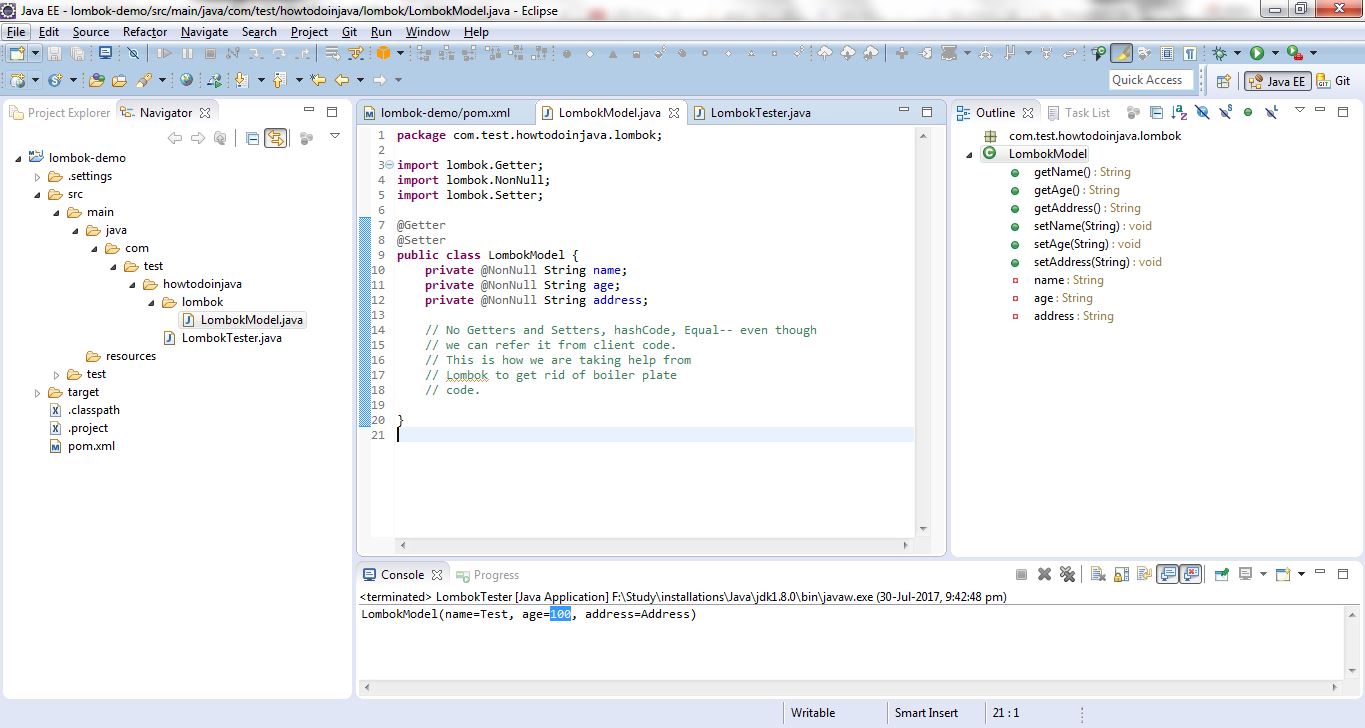
* **Get rid of Setters and Getters**

We all have to generate this java bean pattern heavily in day to day work and it has become so popular that all the IDEs have given feature to generate the Getters and Setters – but once generated by IDE, what’s next? we need to carry those code in whole lifetime of the project and we need to maintain those and it also increases the line of code of the whole project.

With Lombok, we need to add few annotations in the class and we are done. To generate Setters and Getters we need to just add @Getter and @Setter at the class level like this.

|  |
| --- |
| package com.test.javaLive.lombok;    import lombok.Getter;  import lombok.NonNull;  import lombok.Setter;    @Getter  @Setter  public class LombokModel {        private @NonNull String name;      private @NonNull String age;      private @NonNull String address;        // No Getters and Setters, hashCode, Equal-- even though we can refer it from client code.      // This is how we are taking help from Lombok to get rid of boilerplate code.  } |

Look at the outline window on the right, we have all the setters and getters ready to be used.

Getter/Setter generated

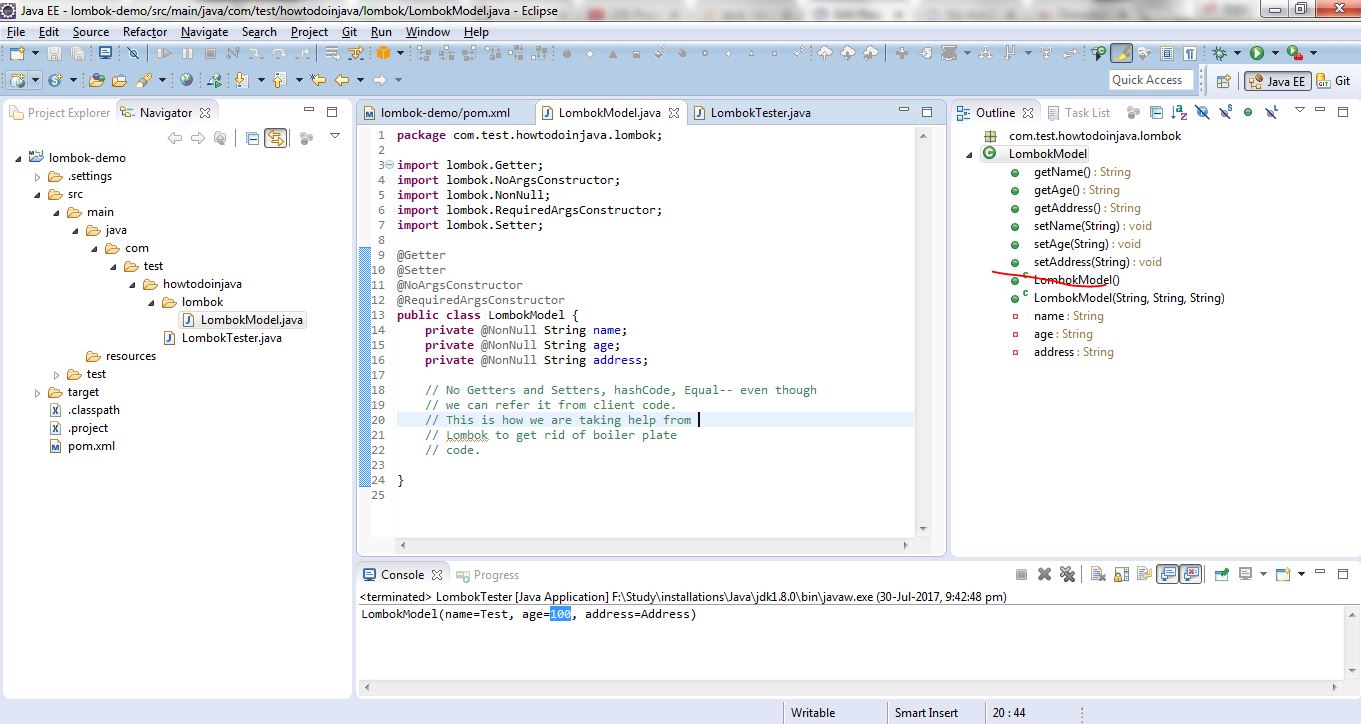
* **Generating Constructors**

Lombok can easily generate the constructors, both no arguments and with fields. We need to add annotation @NoArgsConstructor to generate the implicit no-argument constructor and @RequiredArgsConstructor to add required field constructor.

Also, to make a field required, we need to add @NonNull in the field itself like below.

|  |
| --- |
| package com.test.javaLive.lombok;    import lombok.Getter;  import lombok.NoArgsConstructor;  import lombok.NonNull;  import lombok.RequiredArgsConstructor;  import lombok.Setter;    @Getter  @Setter  @NoArgsConstructor  @RequiredArgsConstructor  public class LombokModel {        private @NonNull String name;      private @NonNull String age;      private @NonNull String address;    } |

Also, look at the outline window on the right, Lombok has already added the constructor as we wished.

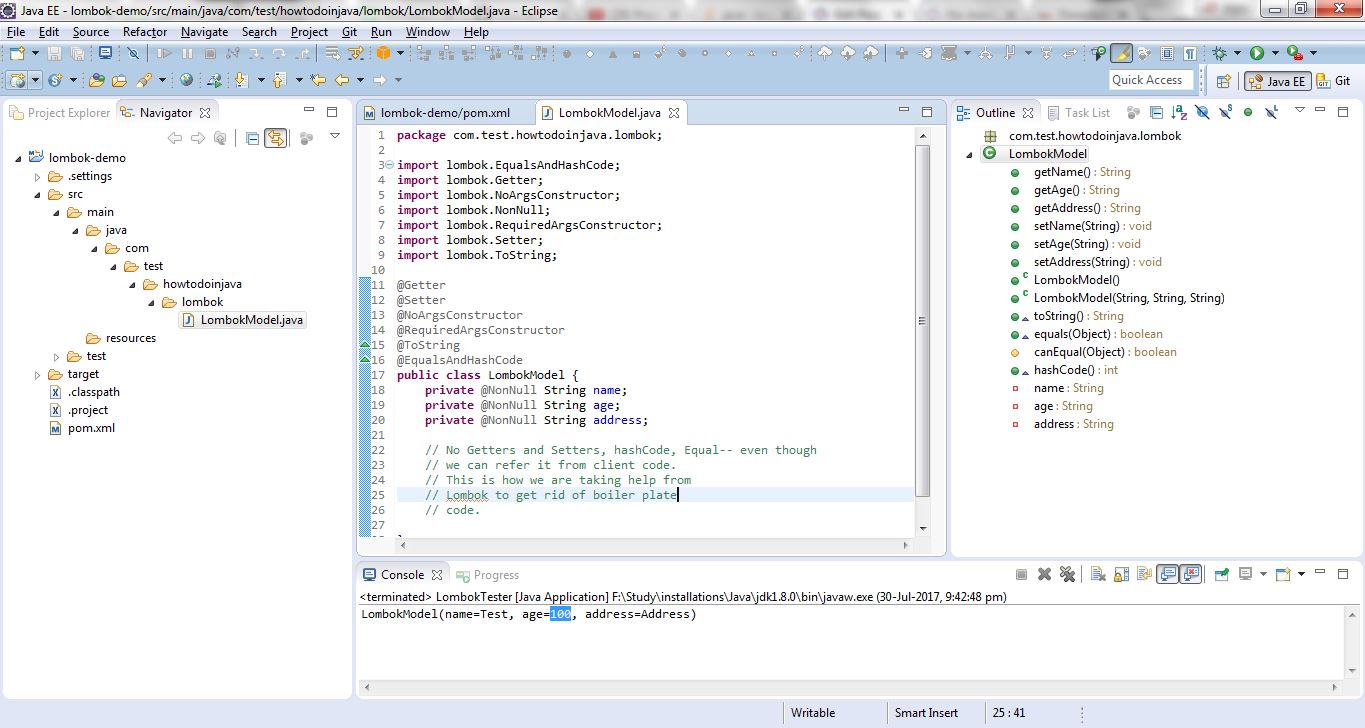
Lombok generated Constructors

* **hashCode(), equals() and toString() methods**

We can easily add default implementation of these methods by adding annotations @ToString and @EqualsAndHashCode in the class level.  
Code will now look like

|  |
| --- |
| package com.test.javaLive.lombok;    import lombok.EqualsAndHashCode;  import lombok.Getter;  import lombok.NoArgsConstructor;  import lombok.NonNull;  import lombok.RequiredArgsConstructor;  import lombok.Setter;  import lombok.ToString;    @Getter  @Setter  @NoArgsConstructor  @RequiredArgsConstructor  @ToString  @EqualsAndHashCode  public class LombokModel {        private @NonNull String name;      private @NonNull String age;      private @NonNull String address;    } |

Also we can see the generated methods by looking at the outline window.

hashCode, equals and toString generated

* **Conventional Object Builder Pattern**

We can easily add [builder pattern](https://howtodoinjava.com/design-patterns/creational/builder-pattern-in-java/) in our code using Lombok. We don’t have to write separate builder class. Lombok will generate the builder along with fluent setter-like methods by simply adding the @Builder annotation in the class level like this.

|  |
| --- |
| package com.test.javaLive.lombok;    import lombok.Builder;  import lombok.EqualsAndHashCode;  import lombok.NonNull;  import lombok.ToString;    @ToString  @EqualsAndHashCode  @Builder  public class LombokModel  {      private @NonNull String name;      private @NonNull String age;      private @NonNull String address;        public static void main(String[] args)      {          LombokModel lombokModel = new LombokModelBuilder()                                      .name("Sajal")                                      .address("India")                                      .age("100")                                      .build();            System.out.println(lombokModel);      }  } |

* **Other Lombok Features**

We have few more annotations, which are also very useful as well. Those are left for you to try and play with. e.g.

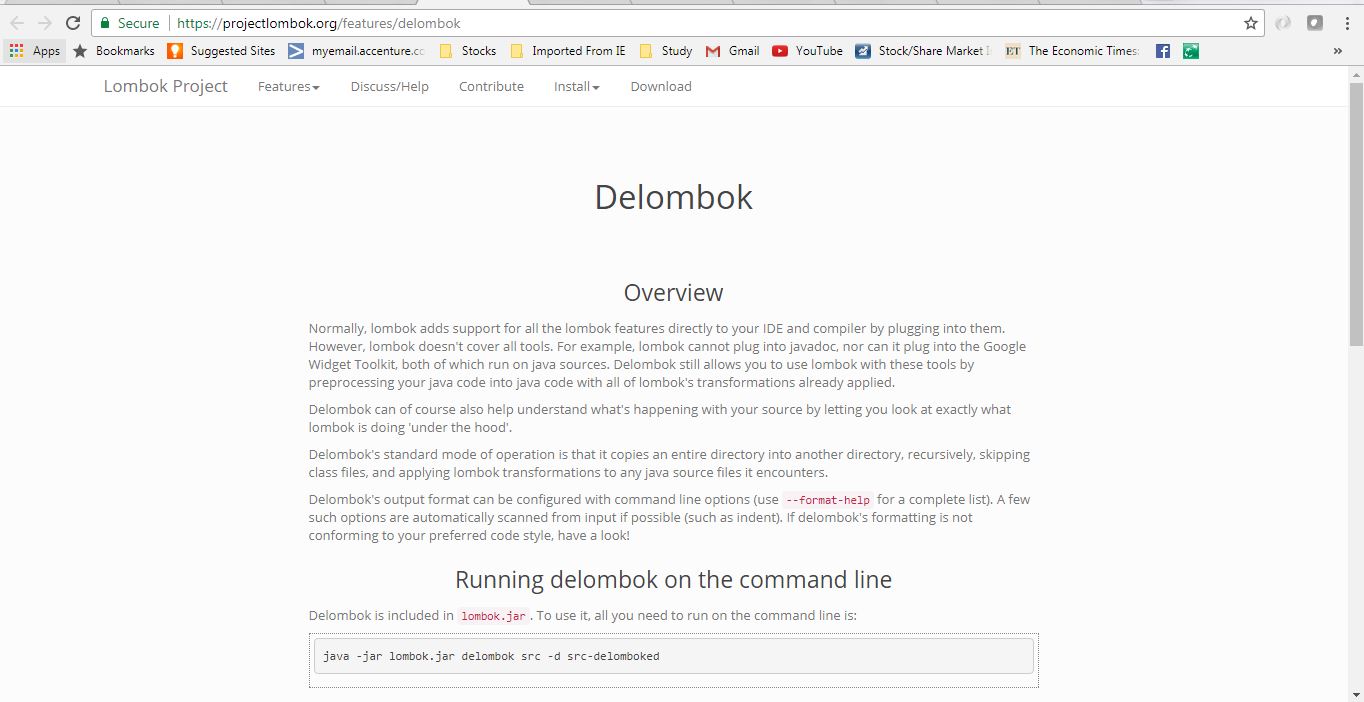
* 1. @Data
  2. @Delegate
  3. @Synchronized
  4. @Slf4j
  5. @Cleanup

**4. Delomboking – Rollback Lombok from Project**

Sometimes all great things are not always accepted by all. Think about this scenario where you had decided to take advantage of Lombok and already added lots of annotations and suddenly due to some change in the decision-makers in the project, you have been asked to stop using Lombok and need to go with the old way of doing things!!!

Now it would not be a easy task at all to revert all the annotations that has been added with the boilerplate code. To do that Lombok itself has provided some steps, by which we can easily replace the annotated source code by the Lombok generated classes.

To get the lombok generated classes, lombok has already documented its steps here. This process is called **delombok**!!!

Delombok Process

To use it, all you need to run on the command line is:

|  |
| --- |
| java -jar lombok.jar delombok src -d src-delomboked |

Above command will duplicate the contents of the src directory into the src-delomboked directory, which will be created if it doesn’t already exist, but delomboked of course.

Delombok tries to preserve your code as much as it can, but comments may move around a little bit, especially comments that are in the middle of a syntax node.

**5. Summary**

In this Lombok tutorial, we saw how we can use Lombok to assist us to get rid of some repetitive code and also it can do some extra things as well. So you can now use this tool whenever applicable and I will suggest you to start using it in your daily work.