

```
@Accessors(fluent=true) @Getter
//@NonNull @Setter
//@NoArgsConstructor
//@AllArgsConstructor
//@RequiredArgsConstructor
//@ToString(exclude = {"lastName"})
//@EqualsAndHashCode(of = {"id"})
@Data
//@Value
@Builder
public class EmployeeLombok {
    private @Setter(AccessLevel.PROTECTED) String id;
    private String firstName;
    private String lastName;
    @NonNull
    private String email;
}
```

Introduction

Lombok is used to reduce boilerplate code for model/data objects, e.g., it can generate getters and setters for those object automatically by using Lombok annotations. The easiest way is to use the @Data annotation.

"Boilerplate" is a term used to describe code that is repeated in many parts of an application with little alteration

Lombok provides a lot of annotations for remove this Boilerplate code.

Like @Data, @Getter, @Setter etc.

Installation

Download install and integrating Lombok jar to IDE

By integrating into the IDE, Project Lombok is able to inject code that is

immediately available to the developer. For example, simply adding the `@Data` annotation to a data class, as below, results in a number of new methods in the IDE.

Installation

Create a maven project and install the dependency.

A jar file will still need to be included in the classpath of any projects that will use Project Lombok annotations.

```
<dependency>  
  <groupId>org.projectlombok</groupId>  
  <artifactId>lombok</artifactId>  
  <version>1.16.16</version>  
  <scope>provided</scope>  
</dependency>
```

Annotations

@Getter and @Setter

1. generate a getter and setter for a field, respectively.
2. Correctly follow convention for boolean properties.

Annotations

@Getter and @Setter

@Getter @Setter

```
private boolean employed = true;
```

```
private boolean employed = true;
```

```
public boolean isEmployed() {
```

```
    return employed;
```

```
}
```

```
public void setEmployed(final boolean  
employed) {
```

```
    this.employed = employed;
```

```
}
```

Annotations

@NonNull

1. Provide fast-fail null check on the corresponding member.
2. NullPointerException will be thrown if Lombok is generating setter and constructor method for your class.

Annotations

@NonNull

@Getter @Setter @NonNull

```
private List<Person>  
members;
```

@NonNull

```
private List<Person>  
members;
```

```
public Family(@NonNull final List<Person>  
members) {  
    if (members == null) throw new  
        java.lang.NullPointerException("members");
```

@NonNull


```
public List<Person> getMembers() { return  
members;}
```

```
public void setMembers(@NonNull final List<Person> members) {  
    if (members == null) throw new  
        java.lang.NullPointerException("members");  
    this.members = members;
```

Annotations

@ToString

1. Implementation of the toString method.
2. By default, any non-static fields will be included
3. Use exclude parameter to exclude any field
4. Set callSuper parameter to true to include output of super class.



```
@ToString(callSuper=true,exclude="someExcludedField")
```

Annotations

@EqualsAndHashCode

1. Generate both equals and hashCode methods
2. Follow equals and hashCode contract.
3. Set callSuper parameter to true to include output of super class.
4. Use exclude parameter to exclude any field

```
@EqualsAndHashCode(callSuper=true,exclude={"address","city","state",  
"zip"})
```

Annotations

@Data

1. Most frequently used annotation.
2. Combines the functionality of @ToString, @EqualsAndHashCode, @Getter and @Setter
3. Generate a constructor.
4. provides everything needed for a Plain Old Java Object (POJO).
5. If a superclass has no default constructor any subclasses cannot use the @Data annotation

Annotations

Some more useful annotations.

1. `@NoArgsConstructor`, `@RequiredArgsConstructor` and `@AllArgsConstructor`
2. `@Cleanup` `@Cleanup` `InputStream in = new FileInputStream(args[0]);`
3. `@Builder`

Benefits

1. Greatly reduce the number of lines of boilerplate code.
2. Reduced maintenance overhead.
3. Fewer bugs and more readable classes.
4. Fast developemt