

lombok

[介绍](#)

[安装lombok插件](#)

[lombok常用注解](#)

[示例](#)

[反编译](#)

lombok

介绍

lombok是一个开源的代码生成库，能以简单的注解形式来简化Java类中的大量样板代码，提高开发人员的开发效率。例如开发中经常需要写的javabean，都需要花时间去添加相应的getter/setter，也许还要去写构造器、equals等方法，而且需要维护，当属性多时会出现大量的getter/setter方法，这些显得很冗长也没有太多技术含量。

lombok能通过注解的方式，在编译时自动为属性生成构造器、getter/setter、equals、hashCode、toString方法，使代码看起来更简洁。

lombok对应的maven坐标：

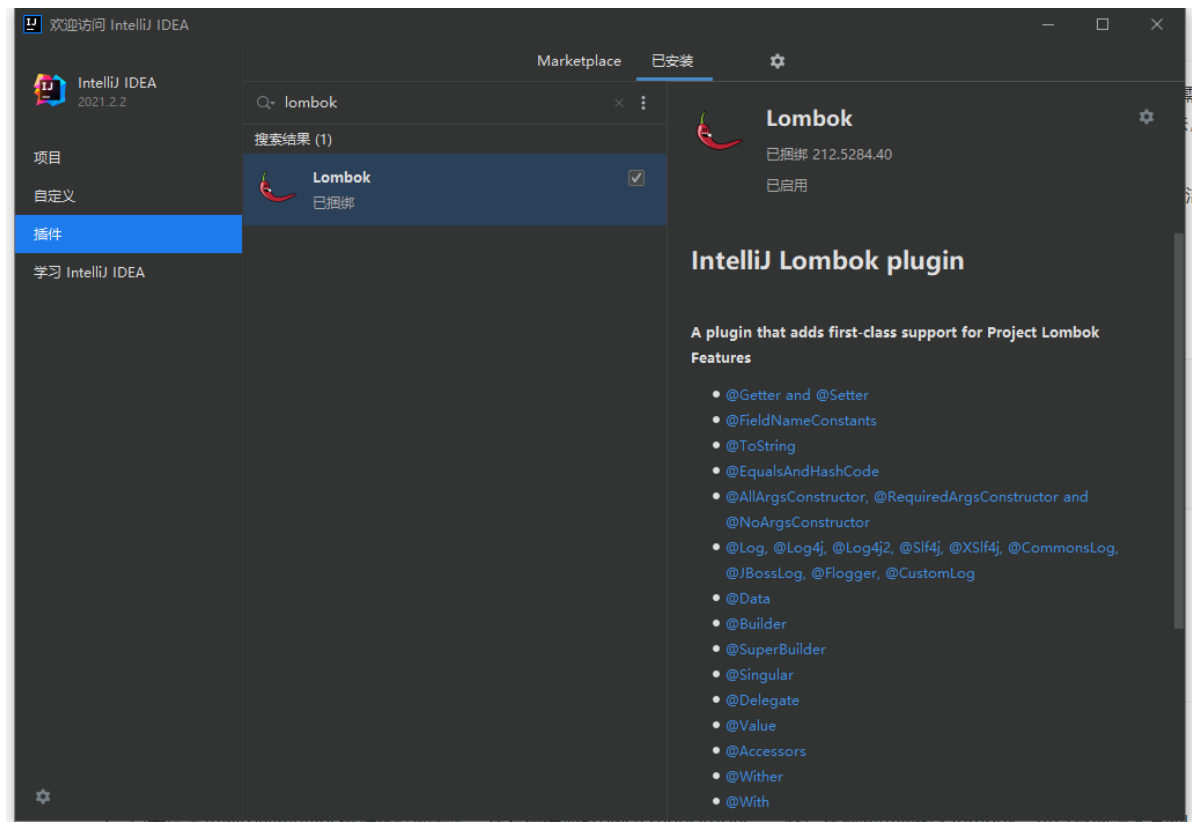
```
1 <dependency>
2   <groupId>org.projectlombok</groupId>
3   <artifactId>lombok</artifactId>
4   <version>1.18.10</version>
5 </dependency>
```

安装lombok插件

要使用lombok需要在IDE中安装对应的lombok插件。本课程使用的开发工具为IntelliJ IDEA，安装插件过程如下：

1、打开IntelliJ IDEA后点击菜单栏中的File-->Settings进入到设置页面

2、点击设置页面中的Plugins进行插件的安装，在右侧选择Browse repositories..., 然后在搜索页面输入lombok，可以查询到下方的Lombok Plugin，鼠标点击Lombok Plugin可在右侧看到Install按钮，点击该按钮便可安装



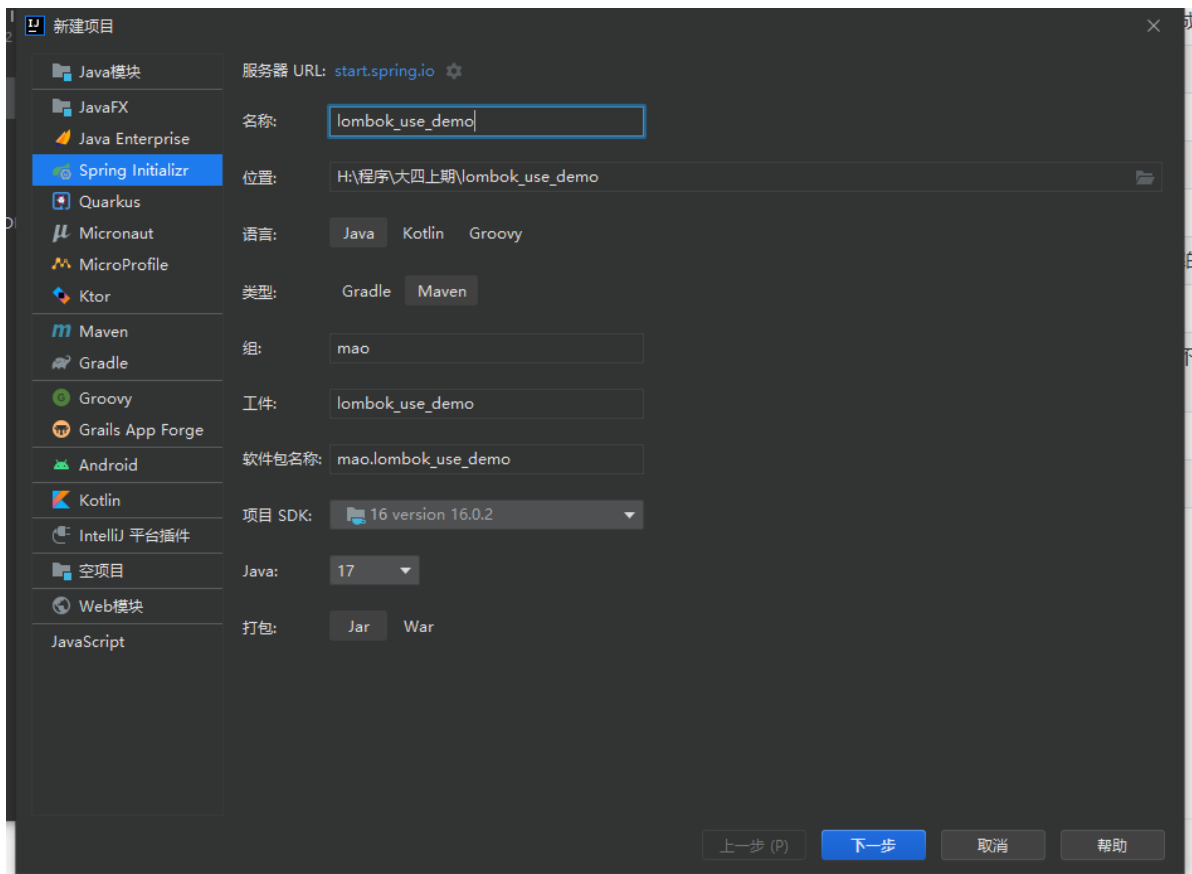
2、点击设置页面中的Plugins进行插件的安装，在右侧选择Browse repositories..., 然后在搜索页面输入lombok，可以查询到下方的L

lombok常用注解

注解	说明
@Setter	注解在类或属性，注解在类时为所有属性生成setter方法，注解在属性上时只为该属性生成setter方法
@Getter	使用方法同@Setter，区别在于生成的是getter方法
@ToString	注解在类，添加toString方法
@EqualsAndHashCode	注解在类，生成hashCode和equals方法
@NoArgsConstructor	注解在类，生成无参的构造方法
@RequiredArgsConstructor	注解在类，为类中需要特殊处理的属性生成构造方法，比如final和被@NonNull注解的属性
@AllArgsConstructor	注解在类，生成包含类中所有属性的构造方法
@Data	注解在类，生成setter/getter、equals、canEqual、hashCode、toString方法，如为final属性，则不会为该属性生成setter方法
@Slf4j	注解在类，生成log变量，用于记录日志
@Builder	将类转变为建造者模式

示例

第一步：创建工程**lombok_use_demo**



第二步：更改pom文件，导入lombok依赖

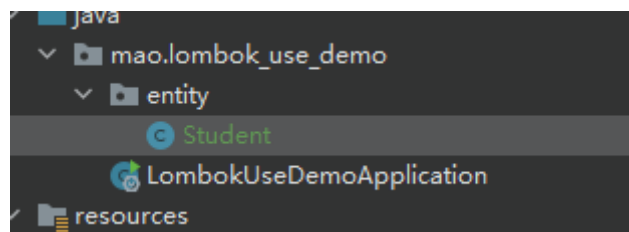
```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
5     https://maven.apache.org/xsd/maven-4.0.0.xsd">
6     <modelVersion>4.0.0</modelVersion>
7
8     <parent>
9         <groupId>org.springframework.boot</groupId>
10        <artifactId>spring-boot-starter-parent</artifactId>
11        <version>2.7.1</version>
12        <relativePath/> <!-- lookup parent from repository -->
13    </parent>
14
15    <groupId>mao</groupId>
16    <artifactId>lombok_use_demo</artifactId>
17    <version>0.0.1-SNAPSHOT</version>
18    <name>lombok_use_demo</name>
19    <description>lombok_use_demo</description>
20
21    <properties>
22        <java.version>11</java.version>
23    </properties>
```

```

23     <dependencies>
24         <dependency>
25             <groupId>org.springframework.boot</groupId>
26             <artifactId>spring-boot-starter</artifactId>
27         </dependency>
28
29         <dependency>
30             <groupId>org.springframework.boot</groupId>
31             <artifactId>spring-boot-starter-test</artifactId>
32             <scope>test</scope>
33         </dependency>
34
35         <!--spring-boot lombok-->
36         <dependency>
37             <groupId>org.projectlombok</groupId>
38             <artifactId>lombok</artifactId>
39             <optional>true</optional>
40         </dependency>
41
42     </dependencies>
43
44     <build>
45         <plugins>
46             <plugin>
47                 <groupId>org.springframework.boot</groupId>
48                 <artifactId>spring-boot-maven-plugin</artifactId>
49             </plugin>
50         </plugins>
51     </build>
52
53 </project>

```

第三步：创建Student实体类



```
public class Student
{
    /**
     * id
     */
    private long id;
    /**
     * 名字
     */
    private String name;
    /**
     * 性
     */
    private String sex;
    /**
     * 年龄
     */
    private int age;
    /**
     * 地址
     */
    private String address;
    /**
     * 电话
     */
    private String phone;
    /**
     * 创建时间
     */
    private Date createTime;
}
```

```
1 package mao.lombok_use_demo.entity;
2
3 import java.util.Date;
4
5 /**
6  * Project name(项目名称): lombok_use_demo
7  * Package(包名): mao.lombok_use_demo.entity
```

```

8  * Class(类名): Student
9  * Author(作者): mao
10 * Author QQ: 1296193245
11 * GitHub: https://github.com/maomao124/
12 * Date(创建日期): 2022/10/25
13 * Time(创建时间): 22:44
14 * Version(版本): 1.0
15 * Description(描述): 无
16 */
17
18 public class Student
19 {
20     /**
21      * id
22      */
23     private long id;
24     /**
25      * 名字
26      */
27     private String name;
28     /**
29      * 性
30      */
31     private String sex;
32     /**
33      * 年龄
34      */
35     private int age;
36     /**
37      * 地址
38      */
39     private String address;
40
41     /**
42      * 电话
43      */
44     private String phone;
45
46     /**
47      * 创建时间
48      */
49     private Date createTime;
50
51 }

```

第四步：加入lombok提供的注解

```

@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class Student
{
    /**

```

```

1  package mao.lombok_use_demo.entity;
2
3  import Lombok.AllArgsConstructor;
4  import Lombok.Builder;
5  import Lombok.Data;
6  import Lombok.NoArgsConstructor;
7
8  import java.util.Date;
9
10 /**
11  * Project name(项目名称): lombok_use_demo
12  * Package(包名): mao.lombok_use_demo.entity
13  * Class(类名): Student
14  * Author(作者): mao
15  * Author QQ: 1296193245
16  * Github: https://github.com/maomao124/
17  * Date(创建日期): 2022/10/25
18  * Time(创建时间): 22:44
19  * Version(版本): 1.0
20  * Description(描述): 无
21  */
22
23 @Data
24 @NoArgsConstructor
25 @AllArgsConstructor
26 @Builder
27 public class Student
28 {
29     /**
30     * id
31     */
32     private long id;
33     /**
34     * 名字
35     */
36     private String name;
37     /**
38     * 性
39     */
40     private String sex;
41     /**
42     * 年龄

```



```
43     */
44     private int age;
45     /**
46      * 地址
47      */
48     private String address;
49
50     /**
51      * 电话
52      */
53     private String phone;
54
55     /**
56      * 创建时间
57      */
58     private Date createTime;
59
60 }
```

第五步：测试

加入日志注解@Slf4j



```
@Test
void test1()
{
    Student student=new Student();
    student.
}
```

	equals(Object o)	boolean
	getAddress()	String
	getAge()	int
	getCreateTime()	Date
	return	return expr
	getId()	long
	getName()	String
	getPhone()	String
	getSex()	String
	hashCode()	int
	setAddress(String address)	void
	setAge(int age)	void

mao.lombok_use_demo.entity.Student
@Contract(value = "null -> false", pure = true)
public boolean equals(Object o)

重写: equals在类Object里边
推断注解: @org.jetbrains.annotations.Contract(value = "null -> false", pure = true)

```
@Test
void test1()
{
    Student student=new Student();
    student.setId(2L);
    student.setName("张三");
    student.setAge(18);
    log
}
```

	Log (= org.slf4j.LoggerFactor ...)	Logger
--	------------------------------------	--------

mao.lombok_use_demo.LombokUseDemoApplicationTests
private static final org.slf4j.Logger log = org.slf4j.LoggerF

测试1:

```
1 package mao.lombok_use_demo;
2
3 import lombok.extern.slf4j.Slf4j;
4 import mao.lombok_use_demo.entity.Student;
5 import org.junit.jupiter.api.Test;
6 import org.springframework.boot.test.context.SpringBootTest;
7
8 @SpringBootTest
9 @Slf4j
10 class LombokUseDemoApplicationTests
11 {
12
13     @Test
14     void contextLoads()
15     {
16     }
17
18     @Test
19     void test1()
20     {
21         Student student=new Student();
22         student.setId(2L);
```

```

23         student.setName("张三");
24         student.setAge(18);
25         log.info(student.toString());
26     }
27 }

```

```

1  2022-10-25 22:52:49.742 INFO 19916 --- [           main]
   m.l.LombokUseDemoApplicationTests : Student(id=2, name=张三, sex=null,
   age=18, address=null, phone=null, createTime=null)

```

测试2:

```

1  package mao.lombok_use_demo;
2
3  import lombok.extern.slf4j.Slf4j;
4  import mao.lombok_use_demo.entity.Student;
5  import org.junit.jupiter.api.Test;
6  import org.springframework.boot.test.context.SpringBootTest;
7
8  @SpringBootTest
9  @Slf4j
10 class LombokUseDemoApplicationTests
11 {
12
13     @Test
14     void contextLoads()
15     {
16     }
17
18     @Test
19     void test1()
20     {
21         Student student = new Student();
22         student.setId(2L);
23         student.setName("张三");
24         student.setAge(18);
25         log.info(student.toString());
26     }
27
28     @Test
29     void test2()
30     {
31         Student student = new Student();
32         student.setId(2L);
33         student.setName("张三");
34         student.setAge(18);
35         log.info(student.getName());
36     }
37 }

```

```
1 2022-10-25 22:53:58.174 INFO 19016 --- [          main]
   m.l.LombokUseDemoApplicationTests      : 张三
```

测试3:

```
1 package mao.lombok_use_demo;
2
3 import lombok.extern.slf4j.Slf4j;
4 import mao.lombok_use_demo.entity.Student;
5 import org.junit.jupiter.api.Test;
6 import org.springframework.boot.test.context.SpringBootTest;
7
8 @SpringBootTest
9 @Slf4j
10 class LombokUseDemoApplicationTests
11 {
12
13     @Test
14     void contextLoads()
15     {
16     }
17
18     /**
19      * test1 , 测试set方法和toString方法
20      */
21     @Test
22     void test1()
23     {
24         Student student = new Student();
25         student.setId(2L);
26         student.setName("张三");
27         student.setAge(18);
28         log.info(student.toString());
29     }
30
31     /**
32      * test2, 测试get方法
33      */
34     @Test
35     void test2()
36     {
37         Student student = new Student();
38         student.setId(2L);
39         student.setName("张三");
40         student.setAge(18);
41         log.info(student.getName());
42     }
43
44     /**
45      * test3 , 测试equals和hashCode方法
46      */
```

```

47     @Test
48     void test3()
49     {
50         Student student1 = new Student();
51         student1.setId(2L);
52         student1.setName("张三");
53         student1.setAge(18);
54
55         Student student2 = new Student();
56         student2.setId(2L);
57         student2.setName("张三");
58         student2.setAge(19);
59
60         Student student3 = new Student();
61         student3.setId(2L);
62         student3.setName("张三");
63         student3.setAge(18);
64
65         log.info("student1 equals student2:" + student1.equals(student2));
66         log.info("student1 equals student3:" + student1.equals(student3));
67
68         log.info("student1 hashCode:" + student1.hashCode());
69         log.info("student2 hashCode:" + student2.hashCode());
70         log.info("student3 hashCode:" + student3.hashCode());
71
72     }
73 }
74

```

```

1  2022-10-25 22:57:09.792 INFO 18892 --- [          main]
   m.l.LombokUseDemoApplicationTests : student1 equals student2:false
2  2022-10-25 22:57:09.792 INFO 18892 --- [          main]
   m.l.LombokUseDemoApplicationTests : student1 equals student3:true
3  2022-10-25 22:57:09.792 INFO 18892 --- [          main]
   m.l.LombokUseDemoApplicationTests : student1 hashCode:1131099724
4  2022-10-25 22:57:09.793 INFO 18892 --- [          main]
   m.l.LombokUseDemoApplicationTests : student2 hashCode:1846024023
5  2022-10-25 22:57:09.793 INFO 18892 --- [          main]
   m.l.LombokUseDemoApplicationTests : student3 hashCode:1131099724

```

测试4:

```

1  package mao.lombok_use_demo;
2
3  import lombok.extern.slf4j.Slf4j;
4  import mao.lombok_use_demo.entity.Student;
5  import org.junit.jupiter.api.Test;
6  import org.springframework.boot.test.context.SpringBootTest;

```

```

7
8 import java.util.Date;
9
10 @SpringBootTest
11 @Slf4j
12 class LombokUseDemoApplicationTests
13 {
14
15     @Test
16     void contextLoads()
17     {
18     }
19
20     /**
21      * test1 , 测试set方法和toString方法
22      */
23     @Test
24     void test1()
25     {
26         Student student = new Student();
27         student.setId(2L);
28         student.setName("张三");
29         student.setAge(18);
30         log.info(student.toString());
31     }
32
33     /**
34      * test2, 测试get方法
35      */
36     @Test
37     void test2()
38     {
39         Student student = new Student();
40         student.setId(2L);
41         student.setName("张三");
42         student.setAge(18);
43         log.info(student.getName());
44     }
45
46     /**
47      * test3 , 测试equals和hashCode方法
48      */
49     @Test
50     void test3()
51     {
52         Student student1 = new Student();
53         student1.setId(2L);
54         student1.setName("张三");
55         student1.setAge(18);
56
57         Student student2 = new Student();
58         student2.setId(2L);
59         student2.setName("张三");
60         student2.setAge(19);
61
62         Student student3 = new Student();
63         student3.setId(2L);
64         student3.setName("张三");

```

```

65         student3.setAge(18);
66
67         log.info("student1 equals student2:" + student1.equals(student2));
68         log.info("student1 equals student3:" + student1.equals(student3));
69
70         log.info("student1 hashCode:" + student1.hashCode());
71         log.info("student2 hashCode:" + student2.hashCode());
72         log.info("student3 hashCode:" + student3.hashCode());
73
74     }
75
76     /**
77      * test4, 测试所有参数的构造方法
78      */
79     @Test
80     void test4()
81     {
82         Student student = new Student(2009L, "李四", "女", 18, null,
83         "188888888888", new Date());
84         log.info(student.toString());
85     }

```

```

1  2022-10-25 23:00:22.367 INFO 16072 --- [           main]
   m.l.LombokUseDemoApplicationTests : Student(id=2009, name=李四, sex=女,
   age=18, address=null, phone=188888888888, createTime=Tue Oct 25 23:00:22 CST
   2022)

```

测试5:

```

1  package mao.lombok_use_demo;
2
3  import lombok.extern.slf4j.Slf4j;
4  import mao.lombok_use_demo.entity.Student;
5  import org.junit.jupiter.api.Test;
6  import org.springframework.boot.test.context.SpringBootTest;
7
8  import java.util.Date;
9
10 @SpringBootTest
11 @Slf4j
12 class LombokUseDemoApplicationTests
13 {
14
15     @Test
16     void contextLoads()
17     {
18     }
19

```

```

20  /**
21   * test1 , 测试set方法和toString方法
22   */
23  @Test
24  void test1()
25  {
26      Student student = new Student();
27      student.setId(2L);
28      student.setName("张三");
29      student.setAge(18);
30      log.info(student.toString());
31  }
32
33  /**
34   * test2, 测试get方法
35   */
36  @Test
37  void test2()
38  {
39      Student student = new Student();
40      student.setId(2L);
41      student.setName("张三");
42      student.setAge(18);
43      log.info(student.getName());
44  }
45
46  /**
47   * test3 , 测试equals和hashCode方法
48   */
49  @Test
50  void test3()
51  {
52      Student student1 = new Student();
53      student1.setId(2L);
54      student1.setName("张三");
55      student1.setAge(18);
56
57      Student student2 = new Student();
58      student2.setId(2L);
59      student2.setName("张三");
60      student2.setAge(19);
61
62      Student student3 = new Student();
63      student3.setId(2L);
64      student3.setName("张三");
65      student3.setAge(18);
66
67      log.info("student1 equals student2:" + student1.equals(student2));
68      log.info("student1 equals student3:" + student1.equals(student3));
69
70      log.info("student1 hashCode:" + student1.hashCode());
71      log.info("student2 hashCode:" + student2.hashCode());
72      log.info("student3 hashCode:" + student3.hashCode());
73
74  }
75
76  /**
77   * test4, 测试所有参数的构造方法

```



```

78     */
79     @Test
80     void test4()
81     {
82         Student student = new Student(2009L, "李四", "女", 18, null,
"18888888888", new Date());
83         log.info(student.toString());
84     }
85
86     /**
87     * test5
88     */
89     @Test
90     void test5()
91     {
92         Student student = Student.builder().id(12L).name("张三").phone("1666666666").build();
93         log.info(student.toString());
94     }
95 }
96

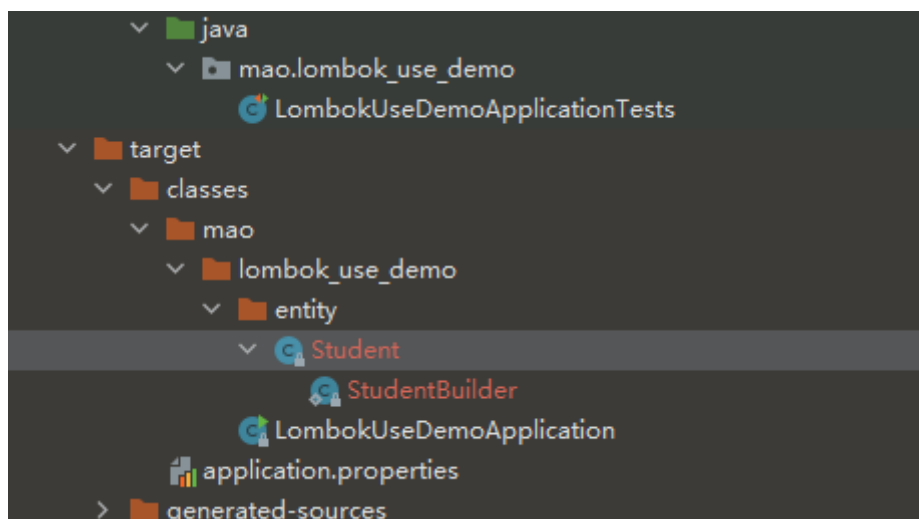
```

```

1  2022-10-25 23:03:05.236 INFO 16048 --- [main]
m.l.LombokUseDemoApplicationTests : Student(id=12, name=张三, sex=null,
age=0, address=null, phone=1666666666, createTime=null)

```

反编译



从target目录里找到对应的实体类，反编译

```
1  //
2  // Source code recreated from a .class file by IntelliJ IDEA
3  // (powered by FernFlower decompiler)
4  //
5
6  package mao.lombok_use_demo.entity;
7
8  import java.util.Date;
9
10 public class Student {
11     private long id;
12     private String name;
13     private String sex;
14     private int age;
15     private String address;
16     private String phone;
17     private Date createTime;
18
19     public static Student.StudentBuilder builder() {
20         return new Student.StudentBuilder();
21     }
22
23     public long getId() {
24         return this.id;
25     }
26
27     public String getName() {
28         return this.name;
29     }
30
31     public String getSex() {
32         return this.sex;
33     }
34
35     public int getAge() {
36         return this.age;
37     }
38
39     public String getAddress() {
40         return this.address;
41     }
42
43     public String getPhone() {
44         return this.phone;
45     }
46
47     public Date getCreateTime() {
48         return this.createTime;
49     }
50 }
```

```

51     public void setId(final long id) {
52         this.id = id;
53     }
54
55     public void setName(final String name) {
56         this.name = name;
57     }
58
59     public void setSex(final String sex) {
60         this.sex = sex;
61     }
62
63     public void setAge(final int age) {
64         this.age = age;
65     }
66
67     public void setAddress(final String address) {
68         this.address = address;
69     }
70
71     public void setPhone(final String phone) {
72         this.phone = phone;
73     }
74
75     public void setCreateTime(final Date createTime) {
76         this.createTime = createTime;
77     }
78
79     public boolean equals(final Object o) {
80         if (o == this) {
81             return true;
82         } else if (!(o instanceof Student)) {
83             return false;
84         } else {
85             Student other = (Student)o;
86             if (!other.canEqual(this)) {
87                 return false;
88             } else if (this.getId() != other.getId()) {
89                 return false;
90             } else if (this.getAge() != other.getAge()) {
91                 return false;
92             } else {
93                 label176: {
94                     Object this$name = this.getName();
95                     Object other$name = other.getName();
96                     if (this$name == null) {
97                         if (other$name == null) {
98                             break label176;
99                         }
100                     } else if (this$name.equals(other$name)) {
101                         break label176;
102                     }
103
104                     return false;
105                 }
106
107                 Object this$sex = this.getSex();
108                 Object other$sex = other.getSex();

```

```

109         if (this$sex == null) {
110             if (other$sex != null) {
111                 return false;
112             }
113         } else if (!this$sex.equals(other$sex)) {
114             return false;
115         }
116
117         label162: {
118             Object this$address = this.getAddress();
119             Object other$address = other.getAddress();
120             if (this$address == null) {
121                 if (other$address == null) {
122                     break label162;
123                 }
124             } else if (this$address.equals(other$address)) {
125                 break label162;
126             }
127
128             return false;
129         }
130
131         label155: {
132             Object this$phone = this.getPhone();
133             Object other$phone = other.getPhone();
134             if (this$phone == null) {
135                 if (other$phone == null) {
136                     break label155;
137                 }
138             } else if (this$phone.equals(other$phone)) {
139                 break label155;
140             }
141
142             return false;
143         }
144
145         Object this$createTime = this.getCreateTime();
146         Object other$createTime = other.getCreateTime();
147         if (this$createTime == null) {
148             if (other$createTime != null) {
149                 return false;
150             }
151         } else if (!this$createTime.equals(other$createTime)) {
152             return false;
153         }
154
155         return true;
156     }
157 }
158
159 protected boolean canEqual(final Object other) {
160     return other instanceof Student;
161 }
162
163
164 public int hashCode() {
165     int PRIME = true;
166     int result = 1;

```

```

167         long $id = this.getId();
168         int result = result * 59 + (int)($id >>> 32 ^ $id);
169         result = result * 59 + this.getAge();
170         Object $name = this.getName();
171         result = result * 59 + ($name == null ? 43 : $name.hashCode());
172         Object $sex = this.getSex();
173         result = result * 59 + ($sex == null ? 43 : $sex.hashCode());
174         Object $address = this.getAddress();
175         result = result * 59 + ($address == null ? 43 :
$address.hashCode());
176         Object $phone = this.getPhone();
177         result = result * 59 + ($phone == null ? 43 : $phone.hashCode());
178         Object $createTime = this.getCreateTime();
179         result = result * 59 + ($createTime == null ? 43 :
$createTime.hashCode());
180         return result;
181     }
182
183     public String toString() {
184         long var10000 = this.getId();
185         return "Student(id=" + var10000 + ", name=" + this.getName() + ",
sex=" + this.getSex() + ", age=" + this.getAge() + ", address=" +
this.getAddress() + ", phone=" + this.getPhone() + ", createTime=" +
this.getCreateTime() + ")";
186     }
187
188     public Student() {
189     }
190
191     public Student(final long id, final String name, final String sex,
final int age, final String address, final String phone, final Date
createTime) {
192         this.id = id;
193         this.name = name;
194         this.sex = sex;
195         this.age = age;
196         this.address = address;
197         this.phone = phone;
198         this.createTime = createTime;
199     }
200
201     public static class StudentBuilder {
202         private long id;
203         private String name;
204         private String sex;
205         private int age;
206         private String address;
207         private String phone;
208         private Date createTime;
209
210         StudentBuilder() {
211         }
212
213         public Student.StudentBuilder id(final long id) {
214             this.id = id;
215             return this;
216         }
217

```

```

218     public Student.StudentBuilder name(final String name) {
219         this.name = name;
220         return this;
221     }
222
223     public Student.StudentBuilder sex(final String sex) {
224         this.sex = sex;
225         return this;
226     }
227
228     public Student.StudentBuilder age(final int age) {
229         this.age = age;
230         return this;
231     }
232
233     public Student.StudentBuilder address(final String address) {
234         this.address = address;
235         return this;
236     }
237
238     public Student.StudentBuilder phone(final String phone) {
239         this.phone = phone;
240         return this;
241     }
242
243     public Student.StudentBuilder createTime(final Date createTime) {
244         this.createTime = createTime;
245         return this;
246     }
247
248     public Student build() {
249         return new Student(this.id, this.name, this.sex, this.age,
250             this.address, this.phone, this.createTime);
251     }
252
253     public String toString() {
254         return "Student.StudentBuilder(id=" + this.id + ", name=" +
255             this.name + ", sex=" + this.sex + ", age=" + this.age + ", address=" +
256             this.address + ", phone=" + this.phone + ", createTime=" + this.createTime
257             + ")";
258     }
259 }

```

end

by mao

2022 10 26
