链滴

Lombok 初体验

作者：flowaters

原文链接：https://ld246.com/article/1532190732828

来源网站：链滴

背景

在阅读sharding-sphere源代码时，看到其中使用了Lombok库。同时，在别的两三个源码中，也看了这个库的身影。

Lombok库最大的优点是使用注解来减少重复代码，再也不用写getter, equals等代码了。

Lombok库的官方网站为projectlombok，其源代码托管在github上。

本文来记录一下Lambok库的使用。

安装

eclipse

1. 下载　lombok.jar包
2. 双击jar包：会自动找到eclipse，然后自动安装
3. 重启eclipse
4. 在eclipse的about页面可以看到　Lombok v1.18.0 "Envious Ferret" is installed. https://projectombok.org/，说明安装成功

Jetbrains IntelliJ IDEA

1. Preferences > Plugins
2. Browse repositories...
3. 搜索Lombok Plugin
4. 点击Install plugin
5. 重启IntelliJ IDEA

第一个例子

pom依赖

目前，最新稳定为1.8，引入依赖：

```
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <version>1.18.0</version>
</dependency>
```

类Getter/Setter

```
package testLombok;

import lombok.Getter;
```

```
import lombok.Setter;
import lombok.ToString;

@Getter
@Setter
@ToString
public class Book {

    public String name;

    public String ISBN;

    public String pubDate;

    public static void main(String[] args) {
        Book book = new Book();
        book.setName("note");
        book.setISBN("abeffect");
        book.setPubDate("2018-07-21");
        System.out.println("name: " + book.getName());
        System.out.println(book);
    }
}
```

运行结果

```
name: note
Book(name=note, ISBN=abeffect, pubDate=2018-07-21)
```

字段Getter/Setter

```
package testLombok;

import lombok.Getter;
import lombok.Setter;
import lombok.ToString;

@ToString
public class Book2 {

    public @Setter @Getter String name;

    public @Setter @Getter String ISBN;

    public @Setter @Getter String pubDate;

    public static void main(String[] args) {
        Book2 book = new Book2();
        book.setName("note");
        book.setISBN("abeffect");
        book.setPubDate("2018-07-21");
        System.out.println("name: " + book.getName());
        System.out.println(book);
    }
```

```
}
```

运行结果

```
name: note
Book2(name=note, ISBN=abeffect, pubDate=2018-07-21)
```

Lombok的特征

常用的

- @Getter: 可用在类上，或者单个类字段上。
- @Setter: 可用在类上，或者单个类字段上。
- @ToString
- @EqualsAndHashCode

ToString

- includeFieldNames: 是否包含字段名称
- exclude: 排除字段
- callSuper: 输出父类字段，默认为false

构造函数

- @AllArgsConstructor: 包含所有字段的构造函数，参数顺序与字段定义顺序一致。如果和NotNull解联用，则会检验是否为空。
- @NoArgsConstructor: 无参构造函数
- @RequiredArgsConstructor: 生成包含final和@NotNull字段的构造方法。

@Data

组合注解: @Getter @Setter @RequiredArgsConstructor @ToString @EqualsAndHashCode

日志

- @Log
- @Log4j
- @Log4j2
- @Slf4j
- @CommonsLog
- @JBossLog
- @Flogger: A Fluent Logging API for Java @Google, Google内部的日志系统。
- @XSlf4j: 适用于 追踪应用执行路径, 详情点链接进去看。

示例如下：

```
//@Log
private static final java.util.logging.Logger log = java.util.logging.Logger.getLogger(Book.class
getName());

//@Log4j
private static final org.apache.log4j.Logger log = org.apache.log4j.Logger.getLogger(Book.cla
s);

//@Log4j2
private static final org.apache.logging.log4j.Logger log = org.apache.logging.log4j.LogManag
r.getLogger(Book.class);

//@Slf4j
private static final org.slf4j.Logger log = org.slf4j.LoggerFactory.getLogger(Book.class);

//@CommonsLog
private static final org.apache.commons.logging.Log log = org.apache.commons.logging.Log
actory.getLog(Book.class);

//@JBossLog
private static final org.jboss.logging.Logger log = org.jboss.logging.Logger.getLogger(Book.cl
ss);

//@Flogger
private static final com.google.common.flogger.FluentLogger log = com.google.common.flo
ger.FluentLogger.forEnclosingClass();

//@XSlf4j
private static final org.slf4j.ext.XLogger log = org.slf4j.ext.XLoggerFactory.getXLogger(Book.cl
ss);
```

Builder

bulder 模式构建对象。如上面的示例，可为：

```
package testLombok;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Getter;
import lombok.RequiredArgsConstructor;
import lombok.Setter;
import lombok.ToString;

@Getter
@Setter
@ToString
@AllArgsConstructor
@RequiredArgsConstructor
@Builder
public class Book {
```

```
    public String name;

    public String ISBN;

    public String pubDate;

    public static void main(String[] args) {
        Book book = new Book();
        book.setName("note");
        book.setISBN("abeffect");
        book.setPubDate("2018-07-21");
        System.out.println("name: " + book.getName());
        System.out.println(book);

        Book book2 = Book.builder().name("note").ISBN("abeffect").pubDate("2018-07-21").build
();
        System.out.println(book2);
    }
}
```

结果

```
name: note
Book(name=note, ISBN=abeffect, pubDate=2018-07-21)
Book(name=note, ISBN=abeffect, pubDate=2018-07-21)
```

@Cleanup

自动化关闭流，相当于 jdk1.7 种的 try with resource

示例

```
@Cleanup
InputStream in = new FileInputStream(args[0]);
@Cleanup
OutputStream out = new FileOutputStream(args[1]);
```

默认的清理方法为close，可以通过value来指定不同的清理方法。

类型推导

- var: 可变变量，相当于普通的变量.
- val: 只读变量，相当于final.

@NonNull

变量不能为null，否则报空指针异常。如

```
public NonNullExample(@NonNull Person person) {
    this.name = person.getName();
}
```

对应的为：

```
public NonNullExample(@NonNull Person person) {
    if (person == null) {
        throw new NullPointerException("person");
    }
    this.name = person.getName();
}
```

@Value

用在类上，生成含所有参数的构造方法，get 方法，此外还提供了equals、hashCode、toString 方
。

但是不生成set方法。

@SneakyThrows

将异常包装为RuntimeException，在运行时抛出。如：

```
    @SneakyThrows(UnsupportedEncodingException.class)
    public String byte2String(byte[] bytes) {
        return new String(bytes, "UTF-8");
    }
```

@Synchronized

同步执行，基本相当于在方法前加了synchronized关键字的效果，实际上是使用的内部对象做的同步。

对于非静态方法，使用$lock字段；对于静态方法，使用$LOCK字段。

@Getter(lazy=true)

在实际使用到cached的时候生成cached，同时，Lombok会自动去管理线程安全的问题，不会存在
复赋值的问题。

示例：

```
    @Getter(lazy = true)
    private final double[] cached = expensive();

    private double[] expensive() {
        double[] result = new double[1000000];
        for (int i = 0; i < result.length; i++) {
            result[i] = Math.asin(i);
        }
        return result;
    }
```

experimental

@FieldNameConstants

FieldNameConstants官方文档

从

```
import lombok.experimental.FieldNameConstants;
import lombok.AccessLevel;

@FieldNameConstants
public class FieldNameConstantsExample {
    private final String iAmAField;
    @FieldNameConstants(level = AccessLevel.PACKAGE)
    private final int andSoAmI;
}
```

到

```
public class FieldNameConstantsExample {
    public static final String FIELD_I_AM_A_FIELD = "iAmAField";
    static final String FIELD_AND_SO_AM_I = "andSoAmI";

    private final String iAmAField;
    private final int andSoAmI;
}
```

@Singular

Singular官方文档

@Delegate

Delegate官方文档

@Accessors

Accessors官方文档

@Wither

Wither官方文档

参考

- Lombok experimental features
- 学习Spring Boot：（十五）使用Lombok来优雅的编码
- Lombok 介绍
- Lombok 之 @Getter(lazy=true)
- IntelliJ IDEA Lombok Plugin