

一、集成lombok让代码更简洁

@Data 标签，生成 `getter/setter` `toString()`等方法

@NonNull：让你不在担忧并且爱上 `NullPointerException`

@Cleanup：自动资源管理：不用再在 `finally` 中添加资源的 `close` 方法

@Setter/@Getter：自动生成 `set` 和 `get` 方法

@ToString：自动生成 `toString` 方法

@EqualsAndHashCode：从对象的字段中生成 `hashCode` 和 `equals` 的实现

@NoArgsConstructor/@RequiredArgsConstructor/@AllArgsConstructor

自动生成构造方法

@Data：自动生成 `set/get` 方法，`toString` 方法，`equals` 方法，`hashCode` 方法，不带参数的构造方法

@Value：用于注解 `final` 类

@Builder：产生复杂的构建器 `api` 类

@SneakyThrows：异常处理（谨慎使用）

@Synchronized：同步方法安全的转化

@Getter(lazy=true)：

@Log：支持各种 `logger` 对象，使用时用对应的注解，如：`@Log4`

lombok 底层使用字节码技术ASM修改字节码文件，生成`get`和`set`方法

1、引入依赖

`<dependency>`

`<groupId>org.projectlombok</groupId>`

`<artifactId>lombok</artifactId>`

`</dependency>`

2、@Getter @Setter 这两注解用于自动生成get/set方法

`@Slf4j` 等同于日志 `private static final Logger logger = LoggerFactory.getLogger(当前Controller类.class);`

使用：`log.info(.....);`

3、有可能还要安装lombok插件

二、@Async异步执行方法

启动类加上**@EnableAsync**,在Service服务类要执行的异步方法上加上**@Async**注解

那么加了**@Async**注解的类就是一个异步调用的方法，底层使用多线程技术。

@Async 相当于这个方法重新开辟了单独线程进行执行，原理是：AOP**@Async**技术在运行时创建一个单独线程进行执行。

```
6
7 @SpringBootApplication
8 @EnableAsync // 开启异步调用
9 public class JspApp {
10
11     public static void main(String[] args)
12         SpringApplication.run(JspApp.class, args);
13 }
14
15
16
```

```
10 public class MemberService {
11
12     // 添加用户的时候，会去发送邮件
13     @Async // 相当于这个方法重新开辟了单独线程进行执行
14     public String addMemberAndEmail() {
15         log.info("2");
16         try {
17             Thread.sleep(5000);
18         } catch (Exception e) {
19             // TODO: handle exception
20         }
21     }
22 }
```