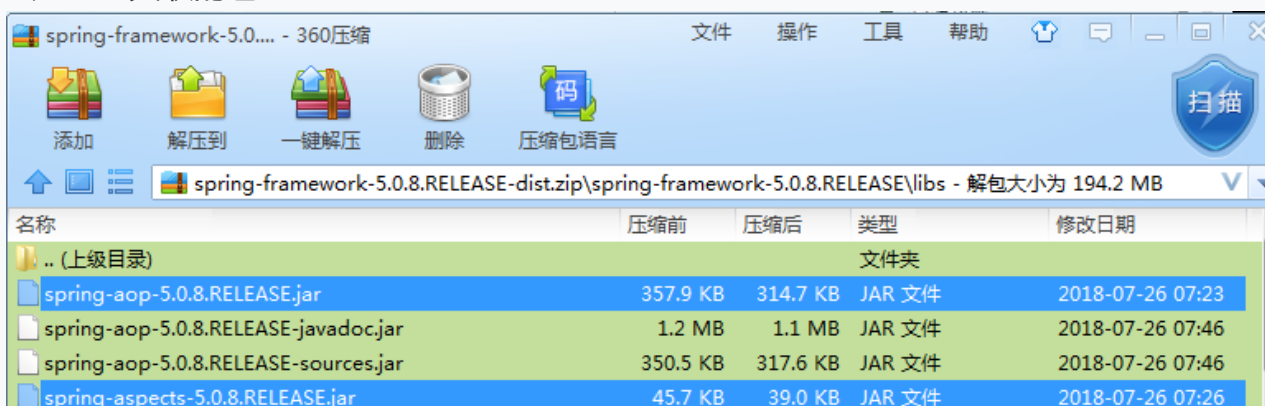


15 Spring_AOP_注解

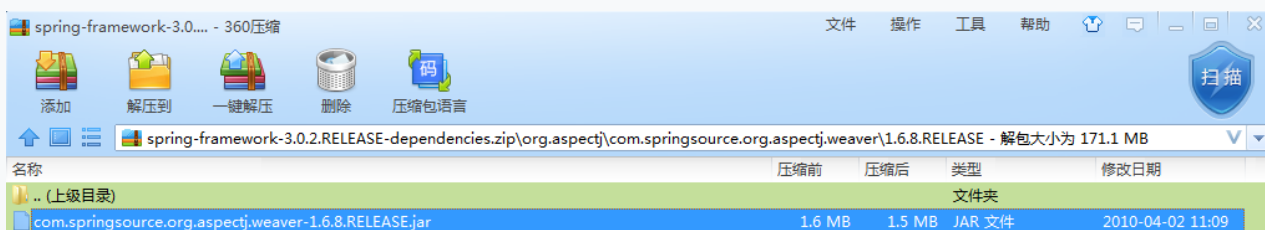
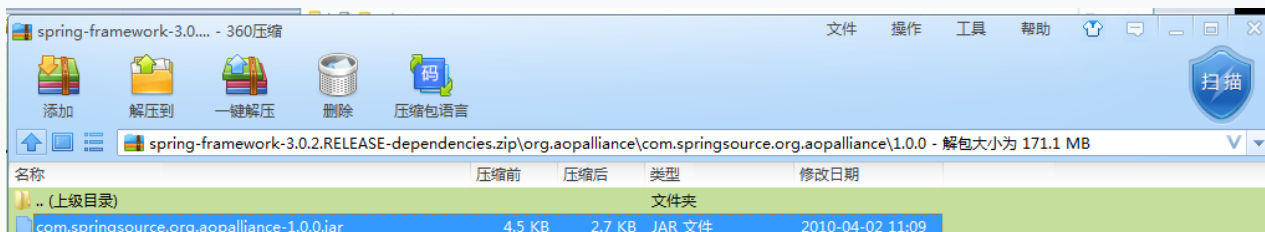
Spring

一：导入jar包

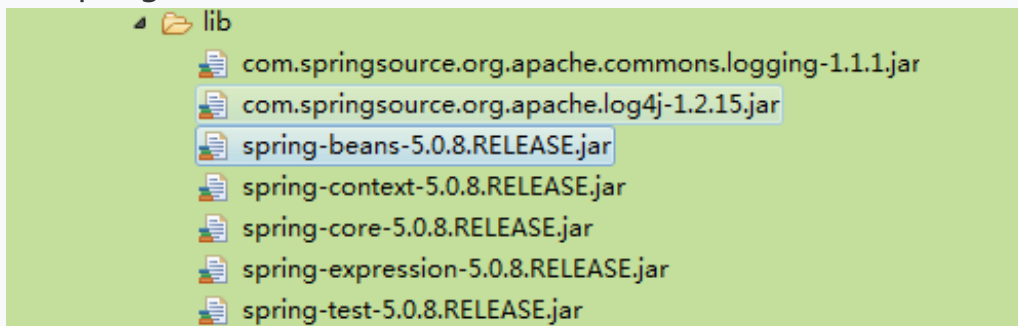
1、AOP关联的包



2、AOP联盟包



3、Spring核心包



二：准备目标对象

```
1. public class UserSerivceImp implements IUserSerivce {
2.     @Override
3.     public void save() throws Exception {
4.         System.out.println("save:查询用户");
5.     }
6.     @Override
7.     public void del() throws Exception {
8.         System.out.println("del:删除用户");
9.     }
10.    @Override
11.    public void upde() throws Exception {
12.        System.out.println("upde:修改用户");
13.    }
14.    @Override
15.    public void add() throws Exception {
16.        System.out.println("add:添加用户");
17.    }
18. }
```

三：注解_方式将通知织入目标对象

1、导入匿名空间

```
1、 xmlns:aop="http://www.springframework.org/schema/aop"
```

```
1. <!--1.配置目标对象 -->
2. <bean name="userSerivceImp"
3.     class="com.spring.serivce.impl.UserSerivceImp"></bean>
4. <!--2.配置通知对象 -->
5. <bean name="usersAdvice" class="com.spring.advice.UsersAdvice"></bean>
6. <!--3.开启注解完成织入 -->
7. <aop:aspectj-autoproxy></aop:aspectj-autoproxy>
```

四：准备通知_注解

- 1、前置通知 |-目标方法运行前调用
- 2、后置通知 |-目标方法运行后调用
- 3、环绕通知 |-目标方法运行前后调用
- 4、异常通知 |-如出现异常就会调用
- 5、最终通知 |- (无论是否出现异常都会调用) 目标方法运行后调用

```
1.  import org.aspectj.lang.ProceedingJoinPoint;
2.  import org.aspectj.lang.annotation.After;
3.  import org.aspectj.lang.annotation.AfterReturning;
4.  import org.aspectj.lang.annotation.AfterThrowing;
5.  import org.aspectj.lang.annotation.Around;
6.  import org.aspectj.lang.annotation.Aspect;
7.  import org.aspectj.lang.annotation.Before;
8.  import org.aspectj.lang.annotation.Pointcut;
9.
10. /**
11.  * @Aspect 表示该类是个通知类
12.  */
13. @Aspect
14. public class UsersAdvice {
15.
16.     @Pointcut("execution(* com.spring.servivce.impl.*SerivceImp.*(..))"
17. )
18.     public void pc() {
19.
20.
21.     /**
22.      * 指定该方法时前置通知 并指定切入点
23.      */
24.     @Before("UsersAdvice.pc()")
25.     public void getBefore() {
26.         System.out.println("--前置通知---");
27.     }
28.
29.
30.     @After("execution(* com.spring.servivce.impl.*SerivceImp.*(..))")
31.     public void getAfter() {
32.         System.out.println("--最终后置通知-无论是否出现异常都会调用--");
33.     }
34.
35.     @Around("execution(* com.spring.servivce.impl.*SerivceImp.*(..))")
```

```

36.     public Object getAround(ProceedingJoinPoint pj) throws Throwable {
37.         System.out.println("--环绕通知-前---");
38.         Object proceed = pj.proceed(); // 调用目标方法的
39.         System.out.println("--环绕通知-后---");
40.         return proceed;
41.     }
42.
43.     @AfterThrowing("execution(* com.spring.servivce.impl.*SerivceImp.*(
44.         ..))")
45.     public void getException() {
46.         System.out.println("--异常通知---");
47.     }
48.
49.     @AfterReturning("execution(* com.spring.servivce.impl.*SerivceImp.*
50.         (..))")
51.     public void getAfterException() {
52.         System.out.println("--后置通知-如出现异常不会调用---");
53.     }
54. }

```

1、测试

```

1.     @RunWith(SpringJUnit4ClassRunner.class)
2.     @ContextConfiguration("classpath:applicationContext.xml")
3.     public class Test {
4.         @Resource(name="userSerivceImp")
5.         private IUserSerivce usi;
6.
7.         @org.junit.Test
8.         public void getVoid02() throws Exception{
9.             usi.save();
10.        }
11.    }

```

Problems @ Javadoc Declaration Console Progress Properties Servers

<terminated> Test.getVoid02 (1) [JUnit] D:\JDK\jdk-8u101-windows-x64\jdk\bin\javaw.exe (2018年8月15日 下午12:59:07)

log4j:WARN No appenders could be found for logger (org.springframework.o
log4j:WARN Please initialize the log4j system properly.

init初始化的方法

- 前置通知---
- 环绕通知-前---

save:查询用户

- 后置通知-无论是否出现异常都会调用--
- 环绕通知-后---
- 后置通知-如出现异常不会调用---

destroy初始化的方法

1、异常通知

在调用的方法上进行促销异常

log4j:WARN Please initialize the log4j system pro

init初始化的方法

- 前置通知---
- 环绕通知-前---
- 后置通知-无论是否出现异常都会调用--
- 异常通知---

destroy初始化的方法