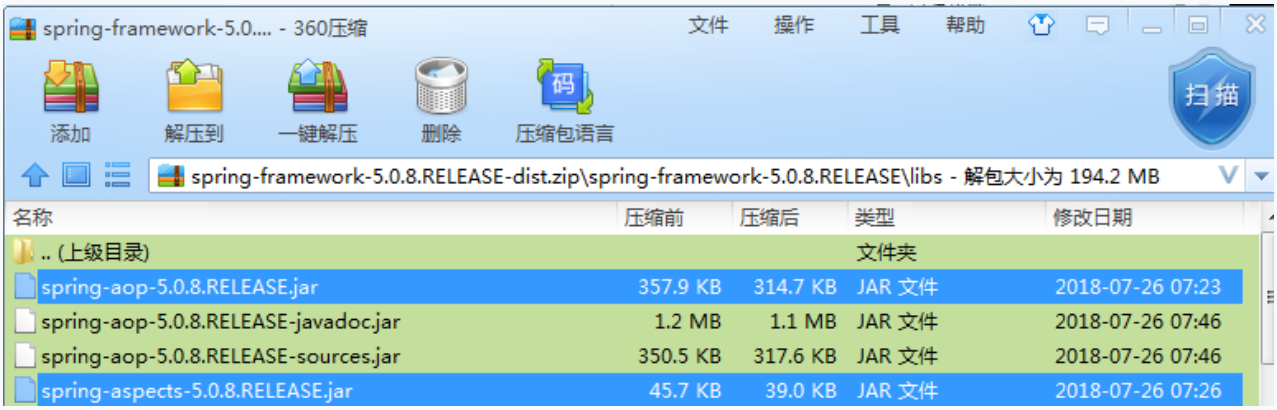
15 Spring_AOP_注解

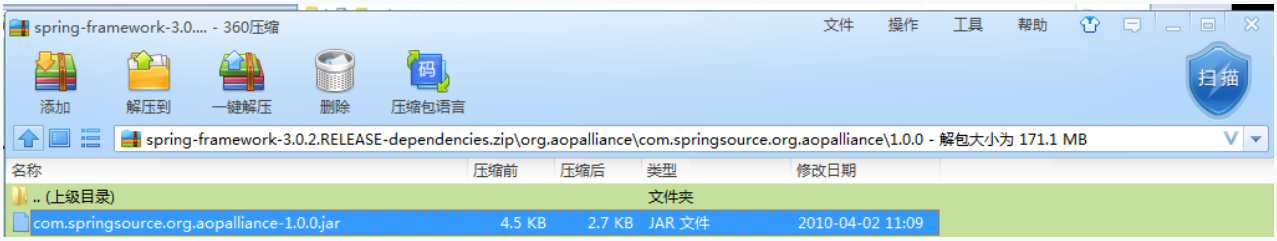
`Spring`

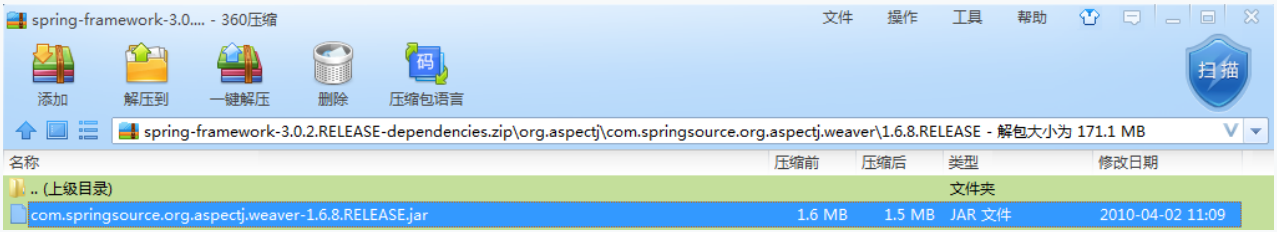
一：导入jar包

1、AOP关联的包

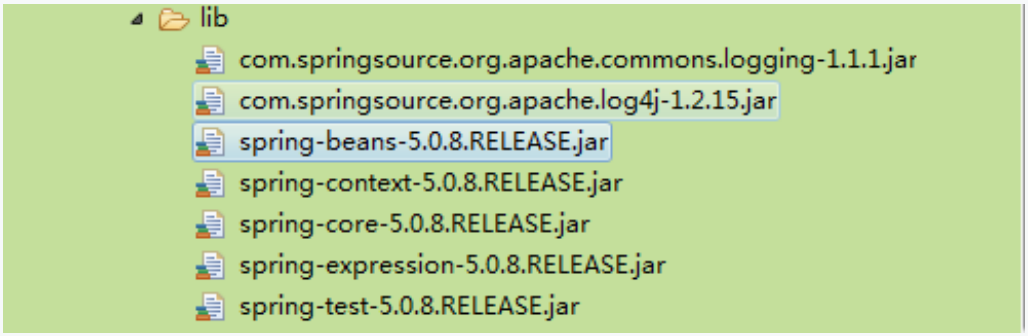


2、AOP联盟包





3、Spring核心包



二：准备目标对象

```
public class UserSerivceImp implements IUserSerivce {
    @Override
    public void save() throws Exception {
        System.out.println("save:查询用户");
    }
    @Override
    public void del() throws Exception {
         System.out.println("del:删除用户");
    }
    @Override
    public void upde() throws Exception {
         System.out.println("upde:修改用户");
    }
    @Override
    public void add() throws Exception {
         System.out.println("add:添加用户");
    }
}
```

三：注解_方式将通知织入目标对象

1、导入冥名空间

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

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

四：准备通知_注解

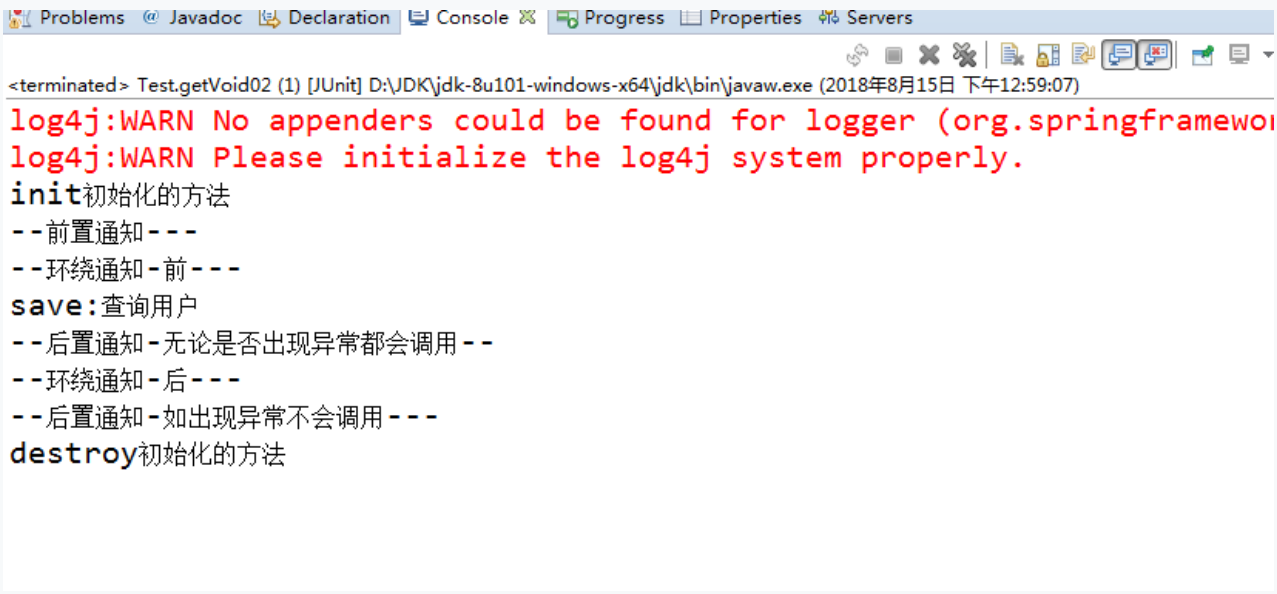
> 1、前置通知 |-目标方法运行前调用
> 2、后置通知 |-目标方法运行后调用
> 3、环绕通知 |-目标方法运行前后调用
> 4、异常通知 |-如出现异常就会调用
> 5、最终通知 |-(无论是否出现异常都会调用) 目标方法运行后调用

```
import org.aspectj.lang.ProceedingJoinPoint;
import org.aspectj.lang.annotation.After;
import org.aspectj.lang.annotation.AfterReturning;
import org.aspectj.lang.annotation.AfterThrowing;
import org.aspectj.lang.annotation.Around;
import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Before;
import org.aspectj.lang.annotation.Pointcut;

/**
 * @Aspect 表示该类是个通知类
 */
@Aspect
public class UsersAdvice {

    @Pointcut("execution(* com.spring.serivce.impl.*SerivceImp.*(..))"
)
    public void pc() {

    }

    /**
     * 指定该方法时前置通知 并指定切入点
     */
    @Before("UsersAdvice.pc()")
    public void getBefore() {
        System.out.println("--前置通知---");
    }


    @After("execution(* com.spring.serivce.impl.*SerivceImp.*(..))")
    public void getAfter() {
        System.out.println("--最终后置通知-无论是否出现异常都会调用--");
    }

    @Around("execution(* com.spring.serivce.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.serivce.impl.*SerivceImp.*(..))")
44.     public void getException() {
45.         System.out.println("--异常通知---");
46.     }
47.
48.     @AfterReturning("execution(* com.spring.serivce.impl.*SerivceImp.*(..))")
49.     public void getAfterException() {
50.         System.out.println("--后置通知-如出现异常不会调用---");
51.     }
52. }
```

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. }
```



1、异常通知

在调用的方法上进行促销异常