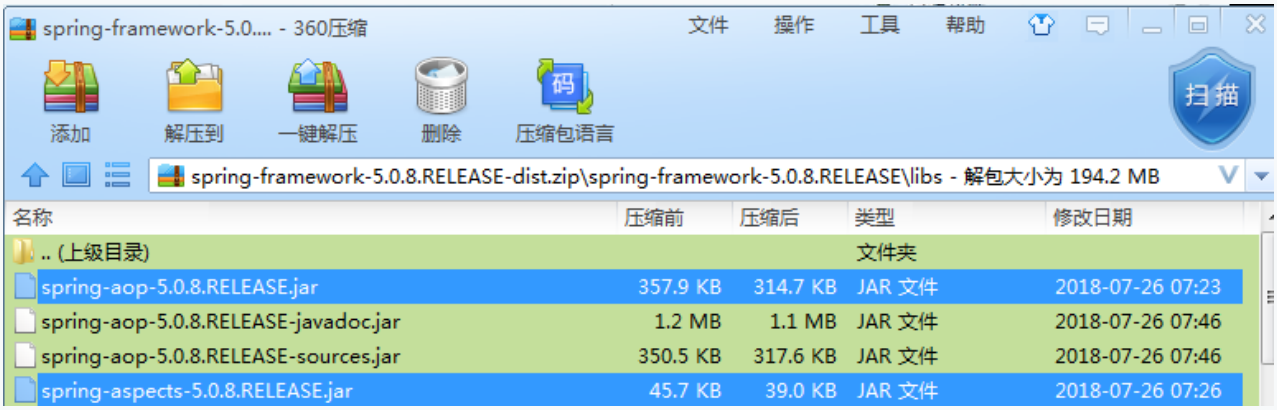
14 Spring_AOP_HelloWord_XML配置

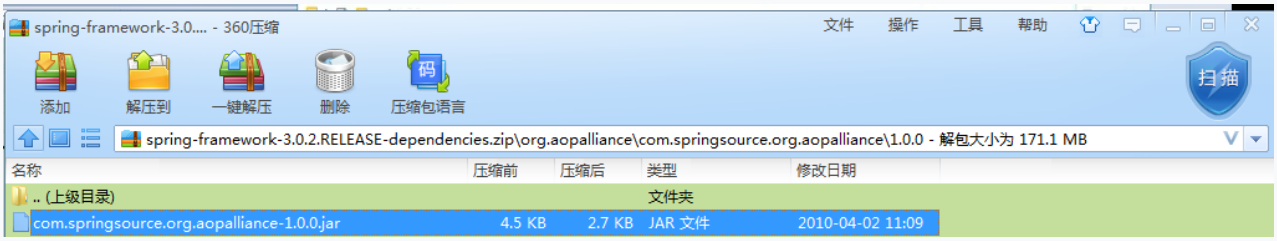
`Spring`

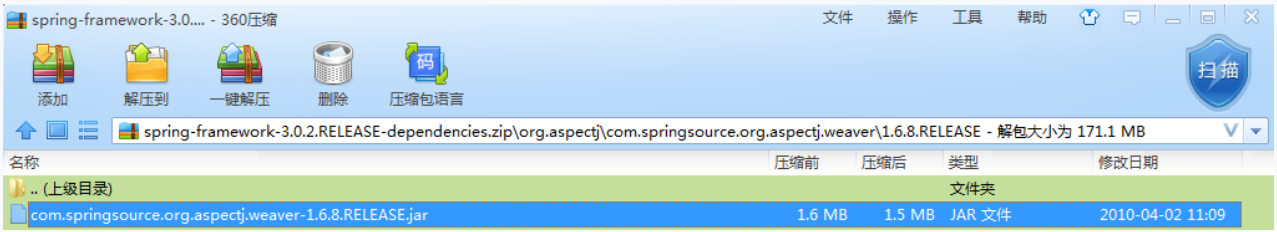
一：导入jar包

1、AOP关联的包

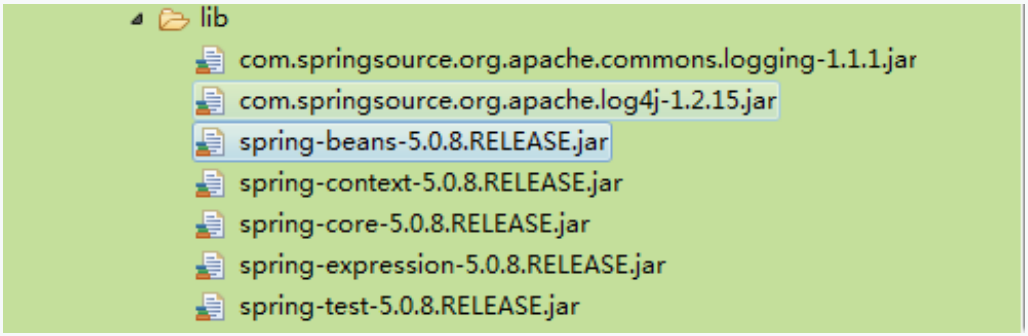


2、AOP联盟包





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：准备通知

> 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

```
public class UsersAdvice {
    public void getBefore() {
        System.out.println("--前置通知---");
    }

    public void getAfter() {
        System.out.println("--最终后置通知-无论是否出现异常都会调用--");
    }

    public Object getAround(ProceedingJoinPoint pj) throws Throwable {
```

```
            System.out.println("--环绕通知-前---");
            Object proceed = pj.proceed();// 调用目标方法的
            System.out.println("--环绕通知-后---");
            return proceed;
        }

        public void getException() {
            System.out.println("--异常通知---");
        }

        public void getAfterException() {
            System.out.println("--后置通知-如出现异常不会调用--");
        }
    }
```

四：配置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"
> 2、通配符
> |-无参数使用()匹配
> |-(..)匹配任意个参数
> |-*() 匹配仅有一个参数的方法*
> /-*(String ,*)匹配仅有2个参数的方法,且第一个参数为String

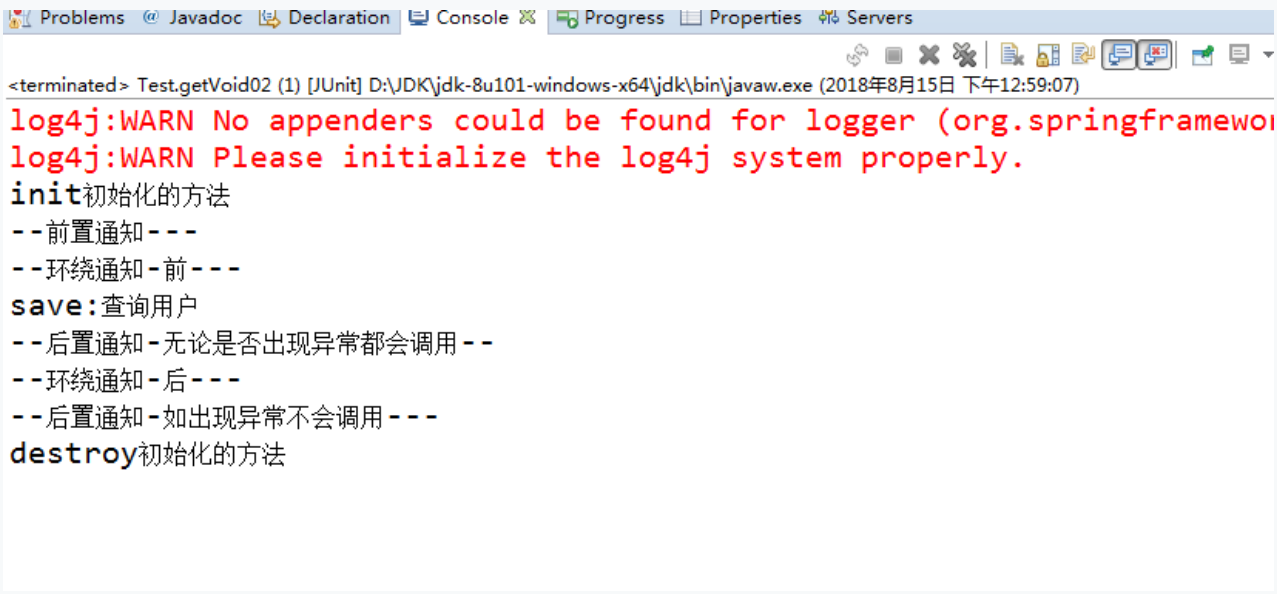
```
<!--1.配置目标对象  -->
<bean name="userSerivceImp"
class="com.spring.serivce.impl.UserSerivceImp"></bean>
<!--2.配置通知对象  -->
<bean name="usersAdvice" class="com.spring.advice.UsersAdvice"></bean>
<!--3.将通知对象织如目标对象    -->
<aop:config>
  <!--配置切入点
         public void com.spring.serivce.impl.UserSerivceImp.save()
      public:可以省略(默认值)
          void com.spring.serivce.impl.UserSerivceImp.save()
          * com.spring.serivce.impl.UserSerivceImp.save()
          * com.spring.serivce.impl.UserSerivceImp.*()
```

```
            * com.spring.serivce.impl.UserSerivceImp.*(..)
            * com.spring.serivce.impl.*SerivceImp.*(..)
            * com.spring.serivce.impl..*SerivceImp.*(..)
    -->
    <aop:pointcut expression="execution(*
com.spring.serivce.impl.*SerivceImp.*(..))" id="pt"/>
    <aop:aspect ref="usersAdvice">
        <!-- 指定前置通知 ;并指定切入点-->
        <aop:before method="getBefore" pointcut-ref="pt"/>
        <!-- 后置通知 -->
        <aop:after-returning method="getAfterException"  pointcut-ref="pt"
/>
        <!-- 环绕通知 -->
        <aop:around method="getAround" pointcut-ref="pt" />
        <!-- 异常拦截 通知 -->
        <aop:after-throwing method="getException" pointcut-ref="pt" />
        <!-- 最终后置 通知 -->
        <aop:after method="getAfter" pointcut-ref="pt"/>
    </aop:aspect>
</aop:config>
```

3、测试

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration("classpath:applicationContext.xml")
public class Test {
    @Resource(name="userSerivceImp")
    private IUserSerivce usi;

    @org.junit.Test
    public  void getVoid02() throws Exception{
        usi.save();
    }
}
```



1、异常通知

在调用的方法上进行促销异常