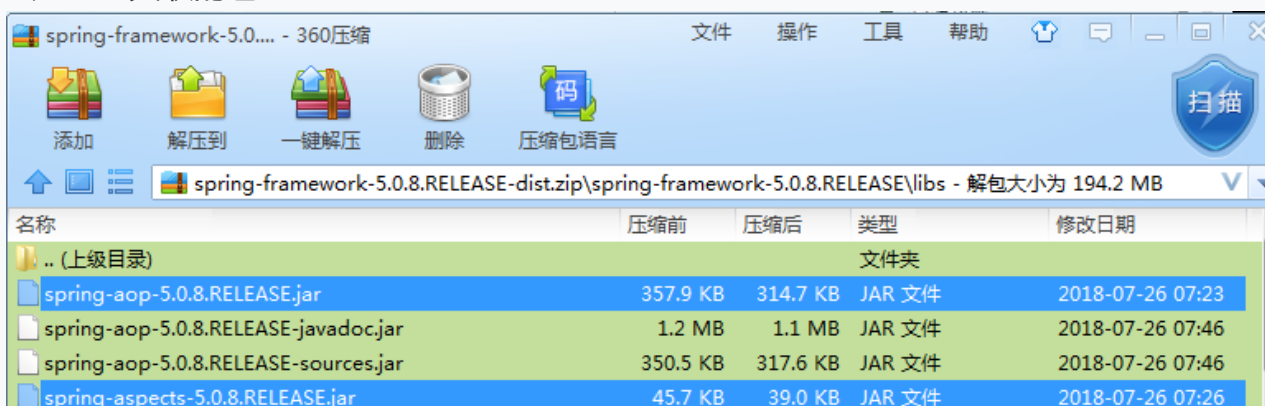


14 Spring_AOP_HelloWord_XML配置

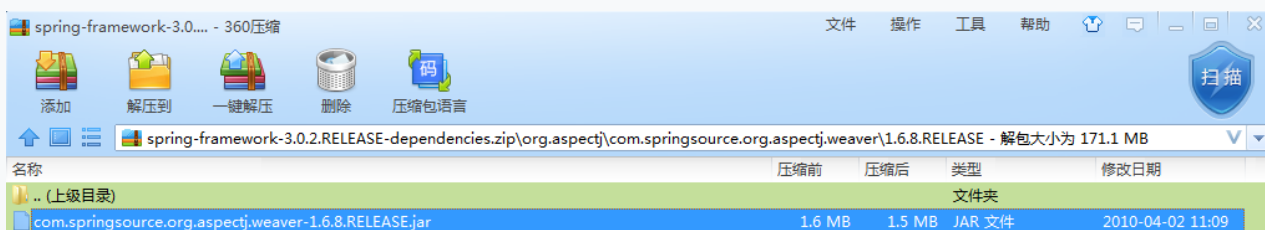
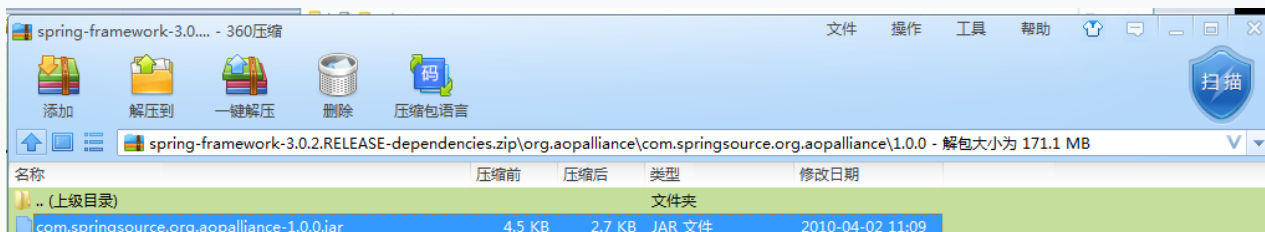
Spring

一：导入jar包

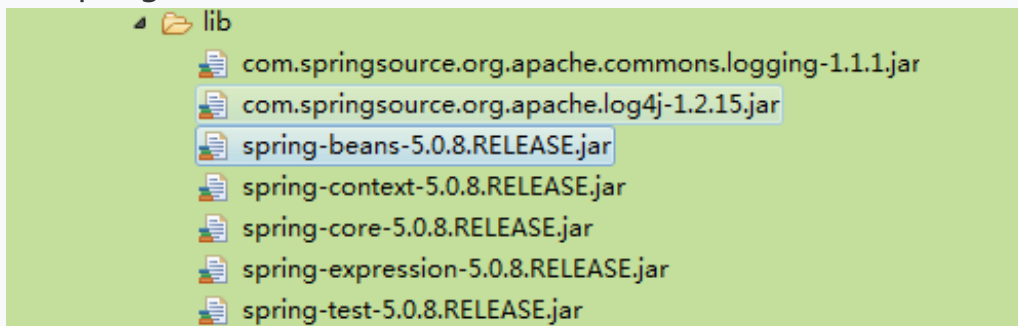
1、AOP关联的包



2、AOP联盟包



3、Spring核心包



二：准备目标对象

```
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：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

```
1.  public class UsersAdvice {
2.      public void getBefore() {
3.          System.out.println("--前置通知---");
4.      }
5.
6.      public void getAfter() {
7.          System.out.println("--最终后置通知-无论是否出现异常都会调用--");
8.      }
9.
10.     public Object getAround(ProceedingJoinPoint pj) throws Throwable {
```

```

11.         System.out.println("--环绕通知-前---");
12.         Object proceed = pj.proceed(); // 调用目标方法的
13.         System.out.println("--环绕通知-后---");
14.         return proceed;
15.     }
16.
17.     public void getException() {
18.         System.out.println("--异常通知---");
19.     }
20.
21.     public void getAfterException() {
22.         System.out.println("--后置通知-如出现异常不会调用--");
23.     }
24. }

```

四：配置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"

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.     <!--1.配置目标对象  -->
2.     <bean name="userServiceImp"
3.         class="com.spring.servive.impl.UserServiceImp"></bean>
4.     <!--2.配置通知对象  -->
5.     <bean name="usersAdvice" class="com.spring.advice.UsersAdvice"></bean>
6.     <!--3.将通知对象织入目标对象  -->
7.     <aop:config>
8.         <!--配置切入点
9.             public void com.spring.servive.impl.UserServiceImp.save()
10.            public:可以省略(默认值)
11.            void com.spring.servive.impl.UserServiceImp.save()
12.            * com.spring.servive.impl.UserServiceImp.save()
13.            * com.spring.servive.impl.UserServiceImp.*()

```

```

13.         * com.spring.servive.impl.UserServiveImp.*(..)
14.         * com.spring.servive.impl.*ServiveImp.*(..)
15.         * com.spring.servive.impl..*ServiveImp.*(..)
16.     -->
17.     <aop:pointcut expression="execution(*
com.spring.servive.impl.*ServiveImp.*(..))" id="pt"/>
18.     <aop:aspect ref="usersAdvice">
19.         <!-- 指定前置通知 ;并指定切入点-->
20.         <aop:before method="getBefore" pointcut-ref="pt"/>
21.         <!-- 后置通知 -->
22.         <aop:after-returning method="getAfterException" pointcut-ref="pt"
/>
23.         <!-- 环绕通知 -->
24.         <aop:around method="getAround" pointcut-ref="pt" />
25.         <!-- 异常拦截 通知 -->
26.         <aop:after-throwing method="getException" pointcut-ref="pt" />
27.         <!-- 最终后置 通知 -->
28.         <aop:after method="getAfter" pointcut-ref="pt"/>
29.     </aop:aspect>
30. </aop:config>

```

3、测试

```

1.     @RunWith(SpringJUnit4ClassRunner.class)
2.     @ContextConfiguration("classpath:applicationContext.xml")
3.     public class Test {
4.         @Resource(name="userServiveImp")
5.         private IUserServive 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