确保AspectJ能用要求，aspectjweaver.jar在1.6.8以上的版本

添加识别AspectJ的两种方式

1.XML中配置

```
<aop:aspectj-autoproxy/>
```

2.java文件中配置

```
@Configuation
@EnableAspectJAutoProxy
public class Config{

}
```

切面

当切面类使用@Aspect注解配置，拥有@Aspect的任何bean被Spring自动是被并应用

用@Aspect注解的类可以有方法和字段，也可以有切入点，通知，和引入声明

@Aspect注解不能通过类路径自动检测发现，需要配合@Comtpnonent注释或xml的bean配置

```
@Aspect
public class 类名{

}
```

切入点

一个切入点是由一个普通方法通过@Pointcut注解生成的，且方法返回值必须为void

```
@Aspect
public class 类名{
 @Pointcut("切入点表达式")
 public void 方法名(){
 方法体
 }
}
```

通知

```
@Aspect
public class 类名{
 @Pointcut("切入点表达式")
 public void pointcut(){
 方法体
 }
 //前置通知
```

```
@Before("可以是切入点表达式，也可以是上面定义过的切入点例如：pointcut()")
public void 方法名(){
 方法体
 }
 //后置通知
 //返回后的通知
 //异常后的通知
 //都与前置通知一样

 //环绕通知
 @Around("切入点表达式")
 public Object 环绕通知方法名(ProceedingJoinPoint pjp){
 //在调用方法之前
 System.out.print("这是在方法之前输出的");
 //指的是具体业务方法执行
 Object 对象 = pjp.proceed();
 return 对象
 //在调用方法之后
 System.out.print("这是在方法之后输出的");
 }

}
```