spring编写代理：半自动

- 让spring创建代理对象，从spring容器中手动的获取代理对象
- 导入jar包：
  - 核心：4+1
  - AOP：AOP联盟（规范）、spring-aop(实现)
- 创建目标类 UserServiceImpl 和 切面类 MyAspect

```
3  public class UserServiceImpl implements UserService {
4
5      public void addUser() {
6
7          System.out.println("b_factory_bean addUser");
8      }
9
10     public void updateUser() {
11
12         System.out.println("b_factory_bean updateUser");
13     }
14
15     public void deleteUser() {
16
17         System.out.println("b_factory_bean deleteUser");
18     }
19
20 }
```

- 切面类 MyAspect要实现MethodInterceptor接口：org.aopalliance.intercept.MethodInterceptor

```
package com.mccken.b_factory_bean;

import org.aopalliance.intercept.MethodInterceptor;
import org.aopalliance.intercept.MethodInvocation;

/*
 * 通知
 * *采用环绕通知MethodInterceptor
 * 切面类中确定通知，需要实现不同接口，接口就是规范，从而就确定方法名称
 */
public class MyAspect implements MethodInterceptor{

    public Object invoke(MethodInvocation invocation) throws Throwable {
        // TODO Auto-generated method stub

        System.out.println("前1");
        //手动执行目标方法
        Object obj = invocation.proceed();  //proceed(继续)：相当于过滤器中通过

        System.out.println("后1");
        return obj;
    }
```

再创建测试类，测试类中getBean拿的是代理类的bean--->proxyServiceId

```java
public class TestFactoryBean {

    @Test
    public void demo01(){
        String xmlPath="com/mccken/b_factory_bean/beans.xml";
        ApplicationContext applicationContext = new ClassPathXmlApplicationContext(xmlPath);
        //获得代理类
        UserService userService = (UserService) applicationContext.getBean("proxyServiceId");
        userService.addUser();
        userService.updateUser();
        userService.deleteUser();

    }
}
```

下面是重点beans.xml

先注册目标类和切面类

```xml
<!-- 1.创建目标类 -->
<bean class="com.mccken.b_factory_bean.UserServiceImpl" id="userServiceId"></bean>

<!-- 2.创建切面类 -->
<bean id="myAspectId" class="com.mccken.b_factory_bean.MyAspect"></bean>
```

再创建代理类

```xml
<bean id="proxyServiceId" class="org.springframework.aop.framework.ProxyFactoryBean">
    <property name="interfaces" value="com.mccken.b_factory_bean.UserService"></property>
    <property name="target" ref="userServiceId"></property>
    <property name="interceptorNames" value="myAspectId"></property>
    <property name="optimize" value="true"></property>
</bean>
```

可选的

代理类方法

```
<!-- 3.创建代理类
    *使用工厂bean FactoryBean,底层调用getObject() 返回特殊bean
    *ProxyFactoryBean用于创建代理工厂bean，生成特殊代理对象
        interfaces:确定接口们
            通过<array>可以设置多个值
            只有一个值时，value=""
        target:确定目标类
        interceptorNames:通知 切面类的名称，类型String[] 如果设置一个值 value
        optimize ：可选，当value="true时，强制使用cglib代理，默认为JDK动态代理
 -->
```

要注意的点

1.切面类实现的接口

MethodInterceptor是 org.aopalliance.intercept.MethodInterceptor;

接口中执行的方法是 invocation.proceed();

2.测试类中applicationContext.getBean拿的bean是代理类的bean id

3. beans.xml中创建的代理类的bean

class="org.springframework.aop.framework.ProxyFactoryBean" 搜索ProxyFactoryBean

注入四个属性：

1.interfaces

目标类的接口们，Class[]型用vlaue 要的是目标类的接口~

2.target

Object型，要的是ref引入目标类的bean id,

3.interceptorNames

String[]型 用value 要的是切面类的bean id

4.optimize

boolean型 可选，当value="true时，强制使用cglib代理，默认为JDK动态代理