

配置约束可以参考以下的html文件



xsd-configuration.html
86.24KB

一，aop概述

- 1, aop: 面向切面（方面）编程，扩展一个功能，不修改源代码实现，而是通过修改或者添加配置文件的方式进行实现。（aspect oriented programing）
- 2, aop采取横向抽取机制，取代了传统纵向继承体系重复性代码（性能监视，事务管理、安全性检查、缓存）。
- 3, spring的aop使用纯java实现，不需要专门的变异过程和类加载器，在运行期通过代理方式向目标类植入增强代码。
- 4, aspect是一个基于java语言的框架，spring2.0开始，spring 的艾欧引入对aspect的支持，aspect扩展了java语言，提供了一个专门的编译器，在编译时提供横向代码植入。

二，aop的底层原理

1，纵向抽取机制



2，采用横向抽取机制

底层使用：动态代理方式实现

aop : 横向抽取机制

底层使用 动态代理方式实现

第一种情况

* 使用jdk动态代理，针对有接口情况

```
public interface Dao {  
    public void add();  
}
```

使用动态代理方式，创建 接口实现类代理对象

```
public class DaoImpl implements Dao {  
    public void add() {  
        //添加逻辑  
    }  
}
```

* 创建和DaoImpl类平级对象
* 这个对象不是真正对象，代理对象，实现和DaoImpl相同的功能

第二种情况 没有接口情况

```
public class User {  
    public void add() {  
  
    }  
}
```

使用cglib动态代理，没有接口情况

//动态代理实现

* 创建User类的子类的代理对象

* 在子类里面调用父类的方法完成增强

三，aop操作相关术语（aop的专业描述）

- 1, joinpoint: 连接点，类里面可以被增强的方法，这些方法就称为连接点。
- 2, pointcut: 切入点，在类里面可以有很多的方法被增强，比如在实操中，只是增强了类里面的部分方法，实际增强的方法称为切入点。
- 3, advice: 通知或者增强，实际增强的逻辑，称为增强或者通知，如扩展日志功能，这个日志功能就称为增强。
 - ①前置通知：在方法之前执行。
 - ②后置通知：在方法之后执行。
 - ③异常通知：方法出现异常之后执行。
 - ④最终通知：在后置通知之后执行。
 - ⑤环绕通知：在方法之前和之后来执行。
- 4, aspect: 切面，把增强应用到具体方法上面，过程称为切面。

- 5, introduction (引介): 是一种特殊的通知, 在不修改类代码的前提下, introduction 可以在运行期为类动态地添加一些方法或field。
- 6, target (目标对象): 代理的目标对象 (要增强的类) (增强方法所在的类)。
- 7, weaving (织入): 是把增强应用到目标的过程, 把advice应用到target的过程。
- 8, proxy (代理): 一个类被aop织入增强后, 就产生一个结果代理类。

四, spring的aop操作 (基于aspectJ的xml方式)

- 1, 在spring里面, 进行aop操作, 使用aspectJ
- 2, aspectJ是一个面向切面的框架, 它扩展了java语言, aspect定义了aop语法所以他有一个专门的编译器用来生成遵守java字节编码规范的class文件。
- 3, aspectJ是一个基于java语言的框架。
- 4, spring2.0以后新增了aspectJ切入点表达式支持。
- 5, @AspectJ是aspect1.5新增功能, 通过JDK5注解技术, 允许直接在bean类中定义切面。
- 6, 新版本spring框架, 建议使用aspectJ方式来开发aop。
- 7, 使用aspectJ需要导入spring aop和aspect相关jar包。
- 8, 在spring里面进行aop操作, 使用aspectJ实现。
 - (1) aspectJ不是spring的一部分, 和spring一起使用进行aop操作。
 - (2) spring2.0以后新增了对aspectJ支持。
- 9, 使用aspectJ实现aop有俩种方式
 - (1) 基于aspectJ的xml配置
 - (2) 基于aspectJ的注解方式

五, spring和aspectJ配合使用具体实现

- 1, 除了导入基本的jar包之外, 还需要导入aop相关的jar包。



aspectjweaver-1.9....s.jar
1.39MB



aopalliance.jar
4.36KB



spring-aspects-5.0...E.jar
40.16KB



spring-aop-5.0.0.R...E.jar
352.16KB

2，使用表达式配置切入点（其中访问修饰符*表示任意类型的修饰符都可以）

（1）切入点：实际增强的方法

（2）表达式：语法：execution (<访问修饰符>?<返回类型><方法名> (<参数>) <异常> >)

①execution(* cn. java. aop. Book. add(..)):表示在cn. java. aop包下的Book类中的add方法就是增强的对象。

②execution(* cn. java. aop. Book. *(..)): 在book类中的所有方法都需要增强

③execution(* *.*(..)): 所有的方法都需要增强。

④execution(* add*(..)): 表示所有以add开头的方法都需要增强。

3，基于aspectJ的配置信息

①被增强类：

```
package cn.java.aop;
public class Book {
    public void add(){
        System.out.println("add.....");
    }
}
```

②增强类：

```
package cn.java.aop;
public class My_Book {
    public void before1(){
        System.out.println("前置增强.....");
    }
}
```

③application.xml中的配置信息

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:aop="http://www.springframework.org/schema/aop"
        xmlns:context="http://www.springframework.org/schema/context"
        xmlns:jdbc="http://www.springframework.org/schema/jdbc"
        xmlns:jee="http://www.springframework.org/schema/jee"
        xmlns:tx="http://www.springframework.org/schema/tx"
        xmlns:util="http://www.springframework.org/schema/util"
        xmlns:task="http://www.springframework.org/schema/task"
        xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-4.0.xsd
```

```

    http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop-4.0.xsd
    http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-4.0.xsd
    http://www.springframework.org/schema/jdbc
http://www.springframework.org/schema/jdbc/spring-jdbc-4.0.xsd
    http://www.springframework.org/schema/jee
http://www.springframework.org/schema/jee/spring-jee-4.0.xsd
    http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-4.0.xsd
    http://www.springframework.org/schema/util
http://www.springframework.org/schema/util/spring-util-4.0.xsd
    http://www.springframework.org/schema/task
http://www.springframework.org/schema/task/spring-task-4.0.xsd"
    default-lazy-init="false">
    <!-- 开启注解扫描,base-package的值就是需要进行注解开发的类所在的包,若只有
    俩三个包的话,可以写cn.java或者cn ( 1 ) 到指定的包中扫描类、方法、属性上是否有注解
    -->
    <context:component-scan base-package="cn.java"></context:component-
scan>
    <!-- 配置对象 -->
    <bean id="book" class="cn.java.aop.Book"></bean>
    <bean id="mybook" class="cn.java.aop.My_Book"></bean>
    <!-- 配置aop配置 -->
    <aop:config proxy-target-class="true">
        <!-- 配置切入点 -->
        <aop:aspect ref="mybook">
            <aop:pointcut expression="execution(* cn.java.aop.Book.*(..))"
id="MypointCut"/>
            <!-- 配置切面 :把增强用到方法上去 -->
            <!-- 配置增强类型 :
            method : 增强类里面使用哪个方法作为前置 -->
            <aop:before method="before1" pointcut-ref="MypointCut" />
        </aop:aspect>
    </aop:config>
</beans>

```

④测试代码

```

package cn.java.aop;
import org.junit.Test;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
public class test {
    @Test
    public void t1() {
        ApplicationContext context = new
ClassPathXmlApplicationContext("application.xml");
    }
}

```

```
        Book book = (Book) context.getBean("book");
        book.add();
    }
}
```