使用 aop 命名空间标签，需要导入 spring-aop j架构：

```
<?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"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
    http://www.springframework.org/schema/aop
    http://www.springframework.org/schema/aop/spring-aop-3.0.xsd ">

   <!-- bean definition & AOP specific configuration -->

</beans>
```

在应用程序的 CLASSPATH 中使用以下 AspectJ 库文件。这些库文件在一个 AspectJ 装置的 ‘lib’ 目录中是可用的，否则可以在 Internet 中下载它们。

- aspectjrt.jar
- aspectjweaver.jar
- aspectj.jar
- aopalliance.jar

声明一个 aspect

使用元素声明，支持的 bean 使用 ref 属性引用：

```
<aop:config>
   <aop:aspect id="myAspect" ref="aBean">
   ...
   </aop:aspect>
</aop:config>
<bean id="aBean" class="...">
...
</bean>
```

声明一个切入点

在哪里进行通知，定义：

```
<aop:config>
   <aop:aspect id="myAspect" ref="aBean">

   <aop:pointcut id="businessService"
      expression="execution(* com.xyz.myapp.service.*.*(..))"/>
   ...
   </aop:aspect>
</aop:config>
<bean id="aBean" class="...">
...
</bean>
```

下面的示例定义了一个名为 “businessService” 的切入点，该切入点将与 com.tutorialspoint 包下的 Student 类中的 getName() 方法相匹配：

```
<aop:config>
   <aop:aspect id="myAspect" ref="aBean">

   <aop:pointcut id="businessService"
      expression="execution(* com.tutorialspoint.Student.getName(..))"/>
    //<aop:advisor advice-ref="myAspect" pointcut-ref="businessService"/> 连接切入点和切入面的线
   ...
   </aop:aspect>
</aop:config>
<bean id="aBean" class="...">
...
</bean>
```

声明Advice

使用 <aop:{ADVICE NAME}> 元素声明五个建议中的任何一个，如下所示：

```
<aop:config>
   <aop:aspect id="myAspect" ref="aBean">
      <aop:pointcut id="businessService"
         expression="execution(* com.xyz.myapp.service.*.*(..))"/>

      <aop:before pointcut-ref="businessService"
         method="doRequiredTask"/>

      <aop:after pointcut-ref="businessService"
         method="doRequiredTask"/>

      <aop:after-returning pointcut-ref="businessService"
         returning="retVal"
         method="doRequiredTask"/>

      <aop:after-throwing pointcut-ref="businessService"
         throwing="ex"
         method="doRequiredTask"/>

      <aop:around pointcut-ref="businessService"
         method="doRequiredTask"/>
   ...
   </aop:aspect>
</aop:config>
<bean id="aBean" class="...">
...
</bean>
```

可以对不同的建议使用相同的 doRequiredTask 或者不同的方法。这些方法将会作为 aspect 模块的一部分来定义。

基于 AOP 的 XML 架构的示例：

Logging.java　　aspect 模块的一个示例，定义了在各个点调用的方法。

```
package com.tutorialspoint;
public class Logging {
   /**
```

```
    * This is the method which I would like to execute
    * before a selected method execution.
    */
   public void beforeAdvice(){
      System.out.println("Going to setup student profile.");
   }
   /**
    * This is the method which I would like to execute
    * after a selected method execution.
    */
   public void afterAdvice(){
      System.out.println("Student profile has been setup.");
   }
   /**
    * This is the method which I would like to execute
    * when any method returns.
    */
   public void afterReturningAdvice(Object retVal){
      System.out.println("Returning:" + retVal.toString() );
   }
   /**
    * This is the method which I would like to execute
    * if there is an exception raised.
    */
   public void AfterThrowingAdvice(IllegalArgumentException ex){
      System.out.println("There has been an exception: " + ex.toString());
   }
}
```

Student.java

```
package com.tutorialspoint;
public class Student {
   private Integer age;
   private String name;
   public void setAge(Integer age) {
      this.age = age;
   }
```

```
   public Integer getAge() {
      System.out.println("Age : " + age );
      return age;
   }
   public void setName(String name) {
      this.name = name;
   }
   public String getName() {
      System.out.println("Name : " + name );
      return name;
   }
   public void printThrowException(){
       System.out.println("Exception raised");
       throw new IllegalArgumentException();
   }
}
```

下面是 MainApp.java 文件的内容：

```
package com.tutorialspoint;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
public class MainApp {
   public static void main(String[] args) {
      ApplicationContext context =
             new ClassPathXmlApplicationContext("Beans.xml");
      Student student = (Student) context.getBean("student");
      student.getName();
      student.getAge();
      student.printThrowException();
   }
}
```

Beans.xml：

```
<?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"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
```

```
    http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
    http://www.springframework.org/schema/aop
    http://www.springframework.org/schema/aop/spring-aop-3.0.xsd ">

   <aop:config>
      <aop:aspect id="log" ref="logging">
         <aop:pointcut id="selectAll"
         expression="execution(* com.tutorialspoint.*.*(..))"/>
         <aop:before pointcut-ref="selectAll" method="beforeAdvice"/>
         <aop:after pointcut-ref="selectAll" method="afterAdvice"/>
         <aop:after-returning pointcut-ref="selectAll"
                              returning="retVal"
                              method="afterReturningAdvice"/>
         <aop:after-throwing pointcut-ref="selectAll"
                             throwing="ex"
                             method="AfterThrowingAdvice"/>
      </aop:aspect>
   </aop:config>

   <!-- Definition for student bean -->
   <bean id="student" class="com.tutorialspoint.Student">
      <property name="name"  value="Zara" />
      <property name="age"  value="11"/>
   </bean>

   <!-- Definition for logging aspect -->
   <bean id="logging" class="com.tutorialspoint.Logging"/>

</beans>
```

输出：

```
Going to setup student profile.
Name : Zara
Student profile has been setup.
Returning:Zara
Going to setup student profile.
Age : 11
```

```
Student profile has been setup.
Returning:11
Going to setup student profile.
Exception raised
Student profile has been setup.
There has been an exception: java.lang.IllegalArgumentException
.....
other exception content
```

定义的在 com.tutorialspoint 中 选择所有方法的 。在一个特殊的方法之前或者之后执行你的建议，通过替换使用真实类和方法名称的切入点定义中的星号（*）来定义你的切入点来缩短你的执行。

```
<?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"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
    http://www.springframework.org/schema/aop
    http://www.springframework.org/schema/aop/spring-aop-3.0.xsd ">

   <aop:config>
   <aop:aspect id="log" ref="logging">
      <aop:pointcut id="selectAll"
      expression="execution(* com.tutorialspoint.Student.getName(..))"/>
      <aop:before pointcut-ref="selectAll" method="beforeAdvice"/>
      <aop:after pointcut-ref="selectAll" method="afterAdvice"/>
   </aop:aspect>
   </aop:config>

   <!-- Definition for student bean -->
   <bean id="student" class="com.tutorialspoint.Student">
      <property name="name"  value="Zara" />
      <property name="age"  value="11"/>
   </bean>
```

```
   <!-- Definition for logging aspect -->
   <bean id="logging" class="com.tutorialspoint.Logging"/>

</beans>
```

输出：

```
Going to setup student profile.
Name : Zara
Student profile has been setup.
Age : 11
Exception raised
.....
other exception content
```

您