Spring 中基于 AOP 的 XML 架构

Spring 中基于 AOP 的 XML 架构

为了在本节的描述中使用 aop 命名空间标签，你需要导入 spring-aop j 架构，如下所述：

```
<?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>
```

你还需要在你的应用程序的 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，否则你可以在 Internet 中下载它们。

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

声明一个 aspect

一个 **aspect** 是使用 元素声明的，支持的 bean 是使用 **ref** 属性引用的，如下所示：

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

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

这里，“aBean” 将被配置和依赖注入，就像前面的章节中你看到的其他的 Spring bean 一样。

声明一个切入点

一个**切入点**有助于确定使用不同建议执行的感兴趣的连接点（即方法）。在处理基于配置的 XML 架构时，切入点将会按照如下所示定义：

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

声明建议

你可以使用 <aop:{ADVICE NAME}> 元素在一个 中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.*.*(..))"/>
      <!-- a before advice definition -->
      <aop:before pointcut-ref="businessService"
         method="doRequiredTask"/>
      <!-- an after advice definition -->
      <aop:after pointcut-ref="businessService"
         method="doRequiredTask"/>
      <!-- an after-returning advice definition -->
      <!--The doRequiredTask method must have parameter named retVal -->
      <aop:after-returning pointcut-ref="businessService"
         returning="retVal"
         method="doRequiredTask"/>
      <!-- an after-throwing advice definition -->
      <!--The doRequiredTask method must have parameter named ex -->
      <aop:after-throwing pointcut-ref="businessService"
         throwing="ex"
         method="doRequiredTask"/>
      <!-- an around advice definition -->
      <aop:around pointcut-ref="businessService"
         method="doRequiredTask"/>
   ...
   </aop:aspect>
</aop:config>
<bean id="aBean" class="...">
...
</bean>
```

你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

为了理解上面提到的基于 AOP 的 XML 架构的概念，让我们编写一个示例，可以实现几个建议。为了在我们的示例中使用几个建议，让我们使 Eclipse IDE 处于工作状态，然后按照如下步骤创建一个 Spring 应用程序：

步骤	描述
1	创建一个名为 *SpringExample* 的项目，并且在所创建项目的 **src** 文件夹下创建一个名为 *com.tutorialspoint* 的包。
2	使用 *Add External JARs* 选项添加所需的 Spring 库文件，就如在 *Spring Hello World Example* 章节中解释的那样。
3	在项目中添加 Spring AOP 指定的库文件 **aspectjrt.jar**，**aspectjweaver.jar** 和 **aspectj.jar**。
4	在 *com.tutorialspoint* 包下创建 Java 类 **Logging**，*Student* 和 *MainApp*。
5	在 **src** 文件夹下创建 Beans 配置文件 *Beans.xml*。
6	最后一步是创建所有 Java 文件和 Bean 配置文件的内容，并且按如下解释的那样运行应用程序。

这里是 **Logging.java** 文件的内容。这实际上是 aspect 模块的一个示例，它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());
   }
}
```

下面是 **Student.java** 文件的内容：

```
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();
   }
}
```

下面是配置文件 **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>
```

一旦你已经完成的创建了源文件和 bean 配置文件，让我们运行一下应用程序。如果你的应用程序一切都正常的话，这将会输出以下消息：

```
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
```

让我们来解释一下上面定义的在 com.tutorialspoint 中 选择所有方法的 。让我们假设一下，你想要在一个特殊的方法之前或者之后执行你的建议，你可以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>
```

如果你想要执行通过这些更改之后的示例应用程序，这将会输出以下消息：

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