声明一个 aspect

```
package org.xyz;
import org.aspectj.lang.annotation.Aspect;
@Aspect
public class AspectModule {
}
```

在 XML 中按照如下进行配置，和其他任何 bean 一样：

```
<bean id="myAspect" class="org.xyz.AspectModule">
   <!-- configure properties of aspect here as normal -->
</bean>
```

声明一个切入点

声明有两个部分：

- 一个切入点表达式决定了我们感兴趣的哪个方法会真正被执行。
- 一个切入点标签包含一个名称和任意数量的参数。方法的真正内容是不相干的，并且实际上它应该是空的。

定义一个名为 ‘businessService’ 的切入点，该切入点将与 com.tutorialspoint 包下的类中可用的每一个方法相匹配：

```
import org.aspectj.lang.annotation.Pointcut;
@Pointcut("execution(* com.xyz.myapp.service.*.*(..))") // expression
private void businessService() {}  // signature
```

定义了一个名为 ‘getname’ 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：

```
import org.aspectj.lang.annotation.Pointcut;
@Pointcut("execution(* com.tutorialspoint.Student.getName(..))")
private void getname() {}
```

声明建议

使用 @{ADVICE-NAME} 注释声明五个建议中的任意一个，假设已经定义了一个切入点标签方法 businessService()：

```
@Before("businessService()")
public void doBeforeTask(){
 ...
}
@After("businessService()")
public void doAfterTask(){
 ...
}
@AfterReturning(pointcut = "businessService()", returning="retVal")
public void doAfterReturnningTask(Object retVal){
  // you can intercept retVal here.
  ...
}
@AfterThrowing(pointcut = "businessService()", throwing="ex")
public void doAfterThrowingTask(Exception ex){
  // you can intercept thrown exception here.
  ...
}
@Around("businessService()")
public void doAroundTask(){
 ...
}
```

你可以为任意一个建议定义你的切入点内联。下面是在建议之前定义内联切入点的一个示例：

```
@Before("execution(* com.xyz.myapp.service.*.*(..))")
public doBeforeTask(){
 ...
}
```

基于 AOP 的 @