In this chapter, you will learn Spring 2.x AOP usage and some advanced AOP topics, such as advice precedence and introduction. The usage of the Spring AOP framework has changed significantly from version 1.x to 2.x. This chapter focuses on the new Spring AOP approach, which enables you to write more powerful and compatible aspects. Moreover, you will learn how to use the AspectJ framework in Spring applications.

In the last chapter, you learned how classic Spring AOP works through a set of proprietary Spring AOP APIs. In Spring version 2.x, you can write your aspects as POJOs with either AspectJ annotations or XML-based configurations in the bean configuration file. As these two types of configurations have the same effect indeed, most of this chapter will focus on AspectJ annotations while describing XML-based configurations for comparison's sake.

Notice that although the usage of Spring AOP has changed, the core implementation technology remains the same: dynamic proxy. Moreover, Spring AOP is backward compatible, so you can continue to use classic Spring advices, pointcuts, and auto proxy creators in Spring 2.x AOP.

As AspectJ is growing into a complete and popular AOP framework, Spring 2.x supports the use of POJO aspects written with AspectJ annotations in its AOP framework. Since AspectJ annotations will be supported by more and more AOP frameworks in the future, your AspectJ-style aspects are more likely to be reused in other AOP frameworks that support AspectJ.

Keep in mind that although you can apply AspectJ aspects in Spring 2.x AOP, this is not the same as using the AspectJ framework. In fact, there are some limitations on the use of AspectJ aspects in Spring AOP, as Spring only allows aspects to apply to beans declared in the IoC container. If you want to apply aspects outside this scope, you have to use the AspectJ framework, which will be introduced at the end of this chapter.

Upon finishing this chapter, you will be able to write POJO aspects to use in the Spring 2.x AOP framework. You should also be able to make use of the AspectJ framework in your Spring applications.
6-1. Enabling AspectJ Annotation Support in Spring

Problem
Spring version 2.x supports the use of POJO aspects written with AspectJ annotations in its AOP framework. But first you have to enable AspectJ annotation support in the Spring IoC container.

Solution
To enable AspectJ annotation support in the Spring IoC container, you only have to define an empty XML element `<aop:aspectj-autoproxy>` in your bean configuration file. Then Spring will automatically create proxies for any of your beans that are matched by your AspectJ aspects.

How It Works
For the sake of a clearer comparison with classic Spring AOP, the calculator sample used in the last chapter will continue to be used. Recall that you defined the following interfaces for your calculator application:

```java
package com.apress.springrecipes.calculator;

public interface ArithmeticCalculator {
    public double add(double a, double b);
    public double sub(double a, double b);
    public double mul(double a, double b);
    public double div(double a, double b);
}

package com.apress.springrecipes.calculator;

public interface UnitCalculator {
    public double kilogramToPound(double kilogram);
    public double kilometerToMile(double kilometer);
}

Then you provided an implementation for each interface with `println` statements to let you know when the methods are executed.

```java
package com.apress.springrecipes.calculator;

public class ArithmeticCalculatorImpl implements ArithmeticCalculator {
    public double add(double a, double b) {
        double result = a + b;
        System.out.println(a + " + " + b + " = " + result);
    }
}
```