Appendix B. Using XML for the context configuration

A long time ago, when I started using Spring, the developers used XML to configure the context and the Spring framework in general. Today, you can only find XML configurations in older applications that are still supported. The developers quit using XML configurations years ago and replaced them with using annotations because of the difficulty of reading the configuration code. While it's true XML has its good points, it's much easier to use annotations for reading and enhancing the app's maintainability. For this reason, I decided not to include XML configurations in this book.

If you're just now starting with Spring, my advice is to learn XML configurations only if you need to maintain an older project and have no other choice. Start by learning the approaches presented in this book. You can apply these skills with any configuration, even XML. The only thing that differs is that you use a different syntax. But it makes no sense to learn these configurations if you'll never see this approach in practice.

To give you a taste of what using XML for configuration means, I'll show you how to add a bean to the Spring context using this old-fashioned way. You can find the example in the project "sq-app2-ex1."

One difference is that with XML, you need to have a separate file in which you'll define the configurations (in fact, they can be multiple files, but I won't get into unnecessary details). I'll name this file "config.xml," and add it to my Maven project's resources folder. The context of this file is presented in the next code snippet:

```
<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
  http://www.springframework.org/schema/beans/spring-beans.xsd">

  <bean id="parrot1" class="main.Parrot">     ❶
```

```
    <property name="name" value="Kiki" />     ❷
  </bean>
</beans>
```

❶ Creates a bean of type Parrot with the identifier parrot1

❷ Sets the value of the parrot's name to Kiki

The `<beans>` tag is the root of this XML file. Inside the root tag, you find the definition of a bean of type `Parrot`. As in the previous snippet, to define a bean you can use another XML tag, `<bean>`, and to give a name to the parrot instance we use the `<property>` XML tag. This is the idea of XML configurations: you use different XML tags to configure specific features. Whatever we'd do using annotations, now you need to use an XML tag.

In the main class, we create an instance representing the Spring context, and we can test that Spring added the bean successfully by referring to the parrot and printing its name in the console. The next code snippet presents the implementation of the main method. I need to use another class to create the instance of the Spring context.

When creating the Spring context instance using the class `ClassPathXmlApplicationContext`, I also need to provide the "config.xml" file's location containing the XML configuration:

```
public class Main {

  public static void main(String[] args) {
    var context = new ClassPathXmlApplicationContext("config.xml");
    Parrot p = context.getBean(Parrot.class);

    System.out.println(p.getName());
  }
}
```

When you run the app, the name you gave to the parrot instance (`Kiki` in my case) in the XML configuration will be printed in the console.