



## 学习目的与要求

本章重点讲解 Spring 开发环境的构建。通过本章的学习，要求读者了解 Spring 的体系结构，掌握 Spring 开发环境的构建。

## 本章主要内容

- ❖ Spring 的体系结构
- ❖ Spring 开发环境的构建
- ❖ 使用 Eclipse 开发 Spring 入门程序

Spring 是当前主流的 Java 企业级应用程序开发框架，为企业级应用开发提供了丰富的功能。掌握 Spring 框架的使用，已是 Java 开发者必备的技能之一。本章将学习如何使用 Eclipse 开发 Spring 入门程序，不过在此之前需要构建 Spring 的开发环境。

## 1.1 Spring 简介

### ▶ 1.1.1 Spring 的由来

Spring 是一个轻量级的 Java 企业级应用程序开发框架，最早由 Rod Johnson 创建，目的是解决企业级应用开发的业务逻辑层和其他各层的耦合问题。它是一个分层的 Java SE/EE full-stack（一站式）轻量级开源框架，为开发 Java 应用程序提供全面的基础架构支持。Spring 负责基础架构，因此 Java 开发者可以专注于应用程序的开发。

Spring Framework 6.0 于 2022 年 11 月正式发布，这是 2023 年及以后新一代框架的开始，包含 OpenJDK 和 Java 生态系统中当前和即将到来的创新。Spring Framework 6.0 作为重大更新，要求使用 Java 17 或更高版本，并且已迁移到 Jakarta EE 9+（在 jakarta 命名空间中取代了以前基于 javax 的 API），以及对其他基础设施的修改。基于这些变化，Spring Framework 6.0 支持最新的 Web 容器，例如 Tomcat 10，以及最新的持久性框架 Hibernate ORM 6.1。这些特性仅可用于 Servlet API 和 JPA 的 jakarta 命名空间变体。

在基础架构方面，Spring Framework 6.0 引入了 Ahead-Of-Time (AOT) 转换的基础以及对 Spring 应用程序上下文的 AOT 转换和相应的 AOT 处理支持的基础，能够事先将应用程序或 JDK 中的字节码编译成机器码。Spring Framework 6.0 中还有许多新功能和改进可用，例如 HTTP 接口客户端、对 RFC 7807 问题详细信息的支持以及 HTTP 客户端的基于 Micrometer 的可观察性。

### ▶ 1.1.2 Spring 的体系结构

Spring 的功能模块被有组织地分散到约 20 个模块中，这些模块分布在核心容器（Core

Container) 层、数据访问/集成 (Data Access/Integration) 层、Web 层、面向切面编程 (Aspect-Oriented Programming, AOP) 模块、植入 (Instrumentation) 模块、消息传输 (Messaging) 和测试 (Test) 模块中, 如图 1.1 所示。

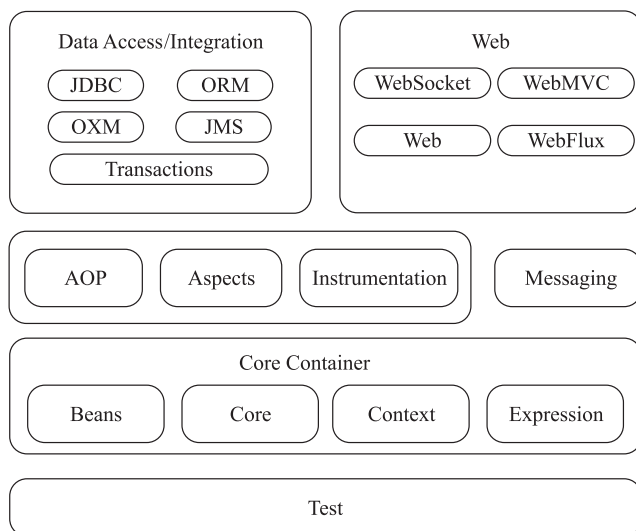


图 1.1 Spring 的体系结构

## ① Core Container 层

Spring 的 Core Container 层是建立其他模块的基础, 由 Beans (spring-beans)、Core (spring-core)、Context (spring-context) 和 Expression (spring-expression, Spring 表达式语言) 等模块组成。

**spring-beans 模块:** 提供了 BeanFactory, 是工厂模式的一个经典实现, Spring 将管理对象称为 Bean。

**spring-core 模块:** 提供了框架的基本组成部分, 包括控制反转 (Inversion of Control, IoC) 和依赖注入 (Dependency Injection, DI) 功能。

**spring-context 模块:** 建立在 spring-beans 和 spring-core 模块的基础之上, 提供一个框架式的对象访问方式, 是访问定义和配置的任何对象的媒介。

**spring-expression 模块:** 提供了一个强大的表达式语言, 用于在运行时查询和操作对象图。这是对 JSP 2.1 规范中规定的统一表达式语言 (Unified Expression Language, UEL) 的扩展。该语言支持设置和获取属性值、属性分配、方法调用、访问数组、集合和索引器的内容、逻辑和算术运算、变量命名以及从 Spring 的 IoC 容器中以名称检索对象。它还支持列表投影、选择以及常见的列表聚合。

## ② AOP 和 Instrumentation 模块

Spring 框架中与 AOP 和 Instrumentation 相关的模块有 AOP (spring-aop) 模块、Aspects (spring-aspects) 模块以及 Instrumentation (spring-instrument) 模块。

**spring-aop 模块:** 提供了一个符合 AOP 要求的面向切面的编程实现, 允许定义方法拦截器和切入点, 将代码按照功能进行分离, 以便干净地解耦。

**spring-aspects 模块:** 提供了与 AspectJ 的集成功能, AspectJ 是一个功能强大且成熟的 AOP 框架。

**spring-instrument 模块：**提供了类植入（Instrumentation）支持和类加载器的实现，可以在特定的应用服务器中使用。Instrumentation 提供了一种虚拟机级别支持的 AOP 实现方式，使得开发者无须对 JDK 做任何升级和改动就可以实现某些 AOP 的功能。

### ③ Messaging 模块

Spring 4.0 以后新增了 Messaging（spring-messaging）模块，该模块提供了对消息传递体系结构和协议的支持。

### ④ Data Access/Integration 层

Data Access/Integration 层由 JDBC（spring-jdbc）、ORM（spring-orm）、OXM（spring-oxm）、JMS（spring-jms）和 Transactions（spring-tx）模块组成。

**spring-jdbc 模块：**提供了一个 JDBC 的抽象层，消除了烦琐的 JDBC 编码和数据库厂商特有的错误代码解析。

**spring-orm 模块：**为流行的对象关系映射（Object-Relational Mapping）API 提供集成层，包括 JPA 和 Hibernate。使用 spring-orm 模块，可以将这些 O/R 映射框架与 Spring 提供的所有其他功能结合使用，例如声明式事务管理功能。

**spring-oxm 模块：**提供了一个支持对象/XML 映射的抽象层实现，例如 JAXB、Castor、JiBX 和 XStream。

**spring-jms 模块（Java Messaging Service）：**指 Java 消息传递服务，包含用于生产和使用消息的功能。自 Spring 4.1 以后，提供了与 spring-messaging 模块的集成。

**spring-tx 模块（事务模块）：**支持用于实现特殊接口和所有 POJO（普通 Java 对象）类的编程和声明式事务管理。

### ⑤ Web 层

Web 层由 Web（spring-web）、WebMVC（spring-webmvc）、WebSocket（spring-websocket）和 WebFlux（spring-webflux）模块组成。

**spring-web 模块：**提供了基本的 Web 开发集成功能。例如多文件上传功能、使用 Servlet 监听器初始化一个 IoC 容器以及 Web 应用上下文。

**spring-webmvc 模块：**也称为 Web-Servlet 模块，包含用于 Web 应用程序的 Spring MVC 和 REST Web Services 实现。Spring MVC 框架提供了领域模型代码和 Web 表单之间的清晰分离，并与 Spring Framework 的所有其他功能集成。本书的第 6 章将会详细讲解 Spring MVC 框架。

**spring-websocket 模块：**Spring 4.0 后新增的模块，它提供了 WebSocket 和 SockJS 的实现，主要是与 Web 前端的全双工通信的协议。

**spring-webflux 模块：**spring-webflux 是一个新的非阻塞的函数式 Reactive Web 框架，可以用来建立异步的、非阻塞的、事件驱动的服务，并且扩展性非常好（该模块是 Spring 5.0 新增的模块）。

### ⑥ Test 模块

Test（spring-test）模块支持使用 JUnit 或 TestNG 对 Spring 组件进行单元测试和集成测试。

## 1.2 Spring 开发环境的构建

在使用 Spring 框架开发 Web 应用之前，应先搭建 Web 应用的开发环境。

扫一扫



视频讲解

## ► 1.2.1 使用 Eclipse 开发 Java Web 应用

为了提高开发效率，通常需要安装 IDE（集成开发环境）工具。Eclipse 是一个可用于开发 Web 应用的 IDE 工具。登录 <http://www.eclipse.org/ide>，选择 Java EE，根据操作系统的位数下载相应的 Eclipse。本书采用的是“eclipse-jee-2022-09-M2-win32-x86\_64.zip”。

在使用 Eclipse 之前，需要对 JDK、Web 服务器和 Eclipse 进行一些必要的配置，因此在安装 Eclipse 之前应该先安装 JDK 和 Web 服务器。

### ① 安装 JDK

安装并配置 JDK（本书采用的 JDK 是 `jdk-18_windows-x64_bin.exe`），在按照提示安装完成 JDK 以后，需要配置环境变量。在 Windows 10 系统下，配置环境变量的示例如图 1.2 和图 1.3 所示。

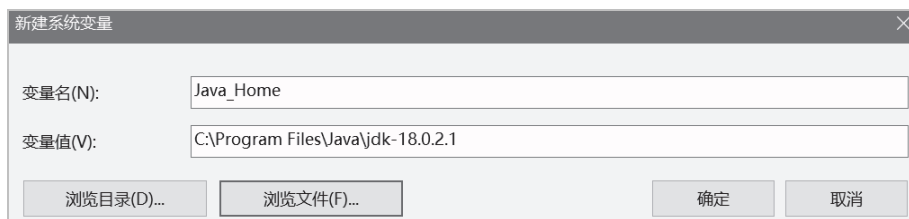


图 1.2 新建系统变量 Java\_Home

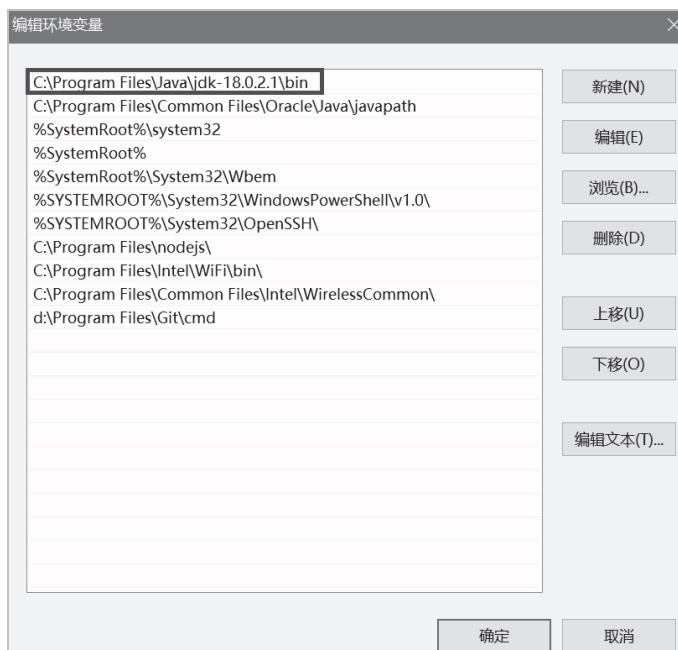


图 1.3 编辑环境变量 Path

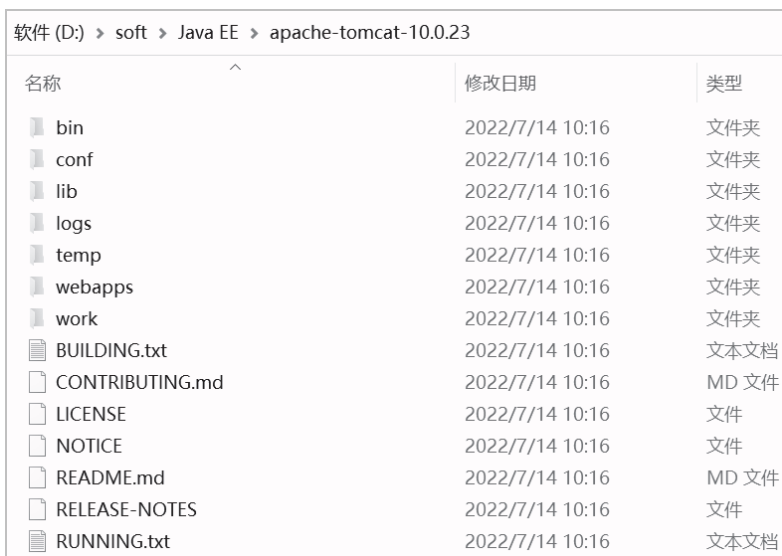
### ② 安装 Web 服务器

目前比较常用的 Web 服务器有 Tomcat、JRun、Resin、WebSphere、WebLogic 等，本书采用的是 Tomcat 10。

登录 Apache 软件基金会的官方网站（<http://jakarta.apache.org/tomcat>），下载 Tomcat 10

的免安装版（本书采用 `apache-tomcat-10.0.23-windows-x64.zip`）。登录网站之后，首先在 Download 中选择 Tomcat 10.0，然后在 Binary Distributions 的 Core 中选择相应版本。

在安装 Tomcat 之前需要先安装 JDK 并配置系统变量 `Java_Home`。将下载的 `apache-tomcat-10.0.23-windows-x64.zip` 解压缩到某个目录下，解压缩后将出现如图 1.4 所示的目录结构。



| 名称              | 修改日期            | 类型    |
|-----------------|-----------------|-------|
| bin             | 2022/7/14 10:16 | 文件夹   |
| conf            | 2022/7/14 10:16 | 文件夹   |
| lib             | 2022/7/14 10:16 | 文件夹   |
| logs            | 2022/7/14 10:16 | 文件夹   |
| temp            | 2022/7/14 10:16 | 文件夹   |
| webapps         | 2022/7/14 10:16 | 文件夹   |
| work            | 2022/7/14 10:16 | 文件夹   |
| BUILDING.txt    | 2022/7/14 10:16 | 文本文档  |
| CONTRIBUTING.md | 2022/7/14 10:16 | MD 文件 |
| LICENSE         | 2022/7/14 10:16 | 文件    |
| NOTICE          | 2022/7/14 10:16 | 文件    |
| README.md       | 2022/7/14 10:16 | MD 文件 |
| RELEASE-NOTES   | 2022/7/14 10:16 | 文件    |
| RUNNING.txt     | 2022/7/14 10:16 | 文本文档  |

图 1.4 Tomcat 目录结构

通过执行 Tomcat 根目录下 `bin` 文件夹中的 `startup.bat` 来启动 Tomcat 服务器。执行 `startup.bat` 启动 Tomcat 服务器会占用一个 MS-DOS 窗口，如果关闭当前 MS-DOS 窗口将关闭 Tomcat 服务器。

Tomcat 服务器启动后，在浏览器的地址栏中输入“`http://localhost:8080`”，将出现如图 1.5 所示的 Tomcat 测试页面。

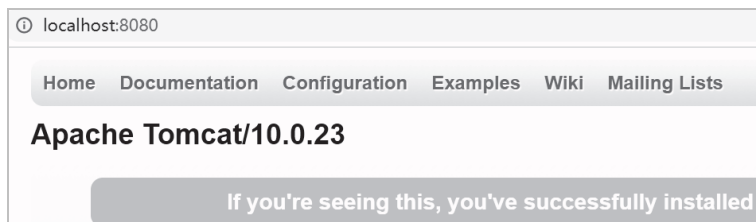


图 1.5 Tomcat 测试页面

### ③ 安装 Eclipse

在将 Eclipse 下载完成后，解压到自己设置的路径下，即可完成安装。安装 Eclipse 后，双击 Eclipse 安装目录下的 `eclipse.exe` 文件启动 Eclipse。

### ④ 集成 Tomcat

启动 Eclipse，选择 `Window/Preferences` 命令，在弹出的 `preferences` 对话框中选择 `Server` 下的 `Runtime Environments` 选项，然后单击 `Add` 按钮，进入如图 1.6 所示的界面，在该界面中可以选择 Tomcat 的版本。

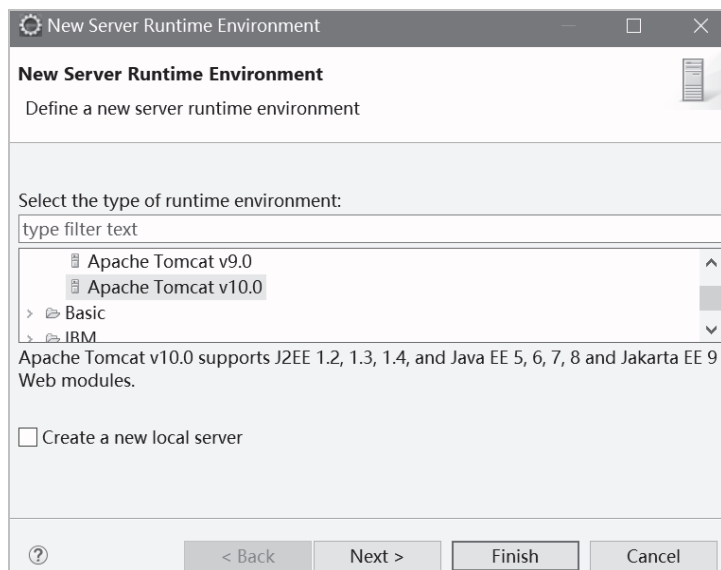


图 1.6 选择 Tomcat 版本的界面

在如图 1.6 所示的界面中选择 Apache Tomcat v10.0 版本，单击 Next 按钮，进入如图 1.7 所示的界面。

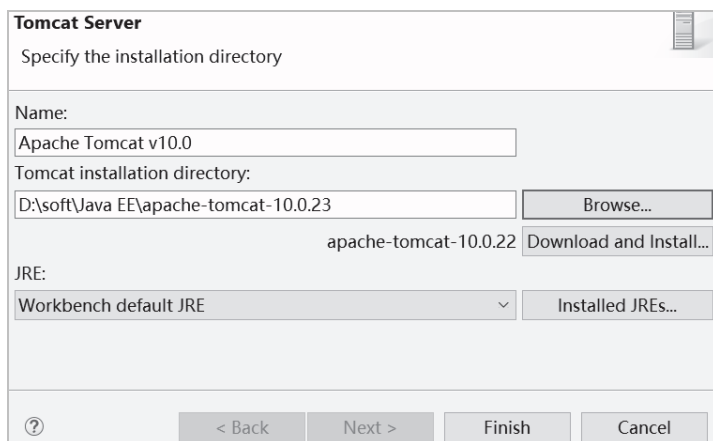


图 1.7 选择 Tomcat 安装目录的界面

在如图 1.7 所示的界面中单击 Browse... 按钮，选择 Tomcat 的安装目录，然后单击 Finish 按钮完成 Tomcat 的集成。

至此，可以使用 Eclipse 创建 Dynamic Web Project，并在 Tomcat 下运行。

## ► 1.2.2 Spring 的下载

在使用 Spring 框架开发应用程序时，需要引用 Spring 框架自身的 JAR 包。Spring Framework 6.0.0 的 JAR 包可以从 Maven 中央库获得。

在 Spring 的 JAR 包中有 4 个基础包，即 spring-core-6.0.0.jar、spring-beans-6.0.0.jar、spring-context-6.0.0.jar 和 spring-expression-6.0.0.jar，分别对应 Spring 核心容器的 spring-core 模块、spring-beans 模块、spring-context 模块和 spring-expression 模块。

对于 Spring 框架的初学者, 在开发 Spring 应用时, 只需要将 Spring 的 4 个基础包和 Spring Commons Logging Bridge 对应的 JAR 包 spring-jcl-6.0.0.jar 复制到 Web 应用的 WEB-INF/lib 目录下即可。

## 1.3 使用 Eclipse 开发 Spring 入门程序

本节通过一个简单的入门程序向读者演示 Spring 框架的使用过程, 具体如下:

### ① 使用 Eclipse 创建 Web 应用并导入 JAR 包

使用 Eclipse 创建一个名为 ch1\_1 的 Dynamic Web Project 应用, 并将 Spring 的 4 个基础包和 Spring Commons Logging Bridge 对应的 JAR 包 spring-jcl-6.0.0.jar 复制到 ch1\_1 的 WEB-INF/lib 目录中, 如图 1.8 所示。



图 1.8 导入 JAR 包

**注意:** 在讲解 Spring MVC 框架之前, 本书的实例并没有真正运行 Web 应用。创建 Web 应用的目的是方便添加相关 JAR 包。

### ② 创建接口 TestDao

Spring 解决的是业务逻辑层和其他各层的耦合问题, 因此它将面向接口的编程思想贯穿整个系统应用。

在 ch1\_1 的 src/main/java 目录下创建一个 dao 包, 并在 dao 包中创建接口 TestDao, 在接口中定义一个 sayHello() 方法, 代码如下:

```
package dao;
public interface TestDao {
    public void sayHello();
}
```

### ③ 创建接口 TestDao 的实现类 TestDaoImpl

在 dao 包下创建 TestDao 的实现类 TestDaoImpl, 代码如下:

```
package dao;
public class TestDaoImpl implements TestDao{
    @Override
    public void sayHello() {
```

扫一扫



视频讲解



```

        System.out.println("Hello, Study hard!");
    }
}

```

#### ④ 创建配置文件 applicationContext.xml

在 ch1\_1 的 src/main/java 目录下创建 Spring 的配置文件 applicationContext.xml，并在该文件中使用实现类 TestDaoImpl 创建一个 id 为 test 的 Bean，代码如下：

```

<?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">
    <!-- 将指定类 TestDaoImpl 配置给 Spring，让 Spring 创建其实例 -->
    <bean id="test" class="dao.TestDaoImpl" />
</beans>

```

**注意：**配置文件的名称可以自定义，但习惯上将其命名为 applicationContext.xml，有时候也命名为 beans.xml。有关 Bean 的创建，将在本书第 3 章中详细讲解，这里读者只需了解即可。另外，配置文件信息不需要读者手写，可以从 Spring 的帮助文档中复制（使用浏览器打开 <https://docs.spring.io/spring-framework/docs/current/reference/html/core.html#spring-core>，在 1.2.1 Configuration Metadata 节中可以找到配置文件的约束信息）。

#### ⑤ 创建测试类

在 ch1\_1 的 src/main/java 目录下创建一个 test 包，并在 test 包中创建 Test 类，代码如下：

```

package test;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import dao.TestDao;
public class Test {
    private static ApplicationContext appCon;
    public static void main(String[] args) {
        appCon = new ClassPathXmlApplicationContext("applicationContext.xml");
        //从容器中获取 test 实例
        TestDao tt = appCon.getBean("test", TestDao.class);
        //test 为配置文件中的 id
        tt.sayHello();
    }
}

```

执行上述 main() 方法后，将在控制台输出“Hello, Study hard!”。在上述 main() 方法中并没有使用 new 运算符创建 TestDaoImpl 类的对象，而是通过 Spring IoC 容器来获取实现类的对象，这就是 Spring IoC 的工作机制。这在本书第 2 章中将详细讲解。

## 1.4 本章小结

本章首先简单介绍了 Spring 的体系结构，然后详细讲解了在 Eclipse 中如何构建 Spring 的开发环境，最后以 ch1\_1 应用为例简要介绍了 Spring 入门程序的开发流程。

扫一扫



自测题

### 习题 1

(1) Spring 的核心容器由哪些模块组成？



(2) 如何找到 Spring 框架的官方 API?

(3) Spring 是一个轻量级的 Java 开发框架，最早由 Rod Johnson 创建，目的是解决企业级应用开发的（ ）和其他各层的耦合问题。

A. 视图层      B. 控制层      C. 数据访问层      D. 业务逻辑层

(4) spring-core 模块提供了 Spring 框架的基本组成部分，包括控制反转和（ ）功能。

A. 依赖注入      B. 切面注入      C. 对象注入      D. 解耦注入



## 学习目的与要求

本章主要介绍了 Spring IoC 的基本概念、Spring IoC 容器以及依赖注入的类型等内容。通过本章的学习，要求读者了解 Spring IoC 容器，掌握 Spring IoC 的基本概念以及依赖注入的类型。

## 本章主要内容

- ❖ Spring IoC 的基本概念
- ❖ Spring IoC 容器
- ❖ 依赖注入的类型

IoC（控制反转）是 Spring 框架的基础，也是 Spring 框架的核心理念。本章将学习 IoC 的基本概念、容器以及依赖注入的类型等内容。

## 2.1 Spring IoC 的基本概念

控制反转（Inversion of Control, IoC）是一个比较抽象的概念，是 Spring 框架的核心，用来削减计算机程序的耦合问题。依赖注入（Dependency Injection, DI）是 IoC 的另外一种说法，是从不同的角度描述相同的概念。下面通过一个实际生活中的例子解释 IoC 和 DI。

当人们需要一件东西时，第一反应就是找东西，比如想吃面包，最直观的做法就是按照自己的口味自己制作面包，也就是主动制作。然而，现在有各种网店、实体店，已经没有必要自己制作面包，如果想吃面包了，到网店或实体店购买即可。

上面只是列举了一个非常简单的例子，但包含了控制反转的思想，即把制作面包的主动权交给店家。下面通过面向对象编程思想继续探讨这两个概念。

当某个 Java 对象（调用者，比如您）需要调用另一个 Java 对象（被调用者，即被依赖对象，比如面包）时，在传统编程模式下，调用者通常会采用“new 被调用者”的代码方式来创建对象（比如您自己制作面包）。这种方式会增加调用者与被调用者之间的耦合性，不利于后期代码的升级与维护。

当 Spring 框架出现后，对象的实例不再由调用者来创建，而是由 Spring 容器（比如网店或实体店）来创建。Spring 容器会负责控制程序之间的关系（比如网店或实体店负责控制您与面包的关系），而不是由调用者的程序代码直接控制。这样，控制权由调用者转移到 Spring 容器，控制权发生了反转，这就是 Spring 的控制反转。

从 Spring 容器角度来看，Spring 容器负责将被依赖对象赋值给调用者的成员变量，相当于为调用者注入它所依赖的实例，这就是 Spring 的依赖注入，主要目的是解耦，体现一种“组合”的理念。

综上所述，控制反转是一种通过描述（在 Spring 中可以是 XML 或注解）并通过第三方去