

# 在线商城系统

## 系列教程

---

Spring

柳曾雄

troy@ujs.edu.cn

2017-3-2

- 一、 Spring
- 二、 Spring 特点
- 三、 Spring 配置
- 四、 Spring 简单使用

## 一、 Spring

春天来了，春暖花开，又到了交配的季节！Spring 的字面意思为‘**春天**’。

如同 Hibernate 的‘**冬天**’一样，字面意思和**框架内涵**没有必然的关系。

那么 Spring 框架到底是干嘛的呢？

Spring 框架是一个**开源**框架，即开发者可以下载源码**学习、使用或者重新发行**（遵守开源协议）。

我们在最前面曾经简单讲过 Spring 框架，现在在这里做个详细的介绍。

Spring 框架最**核心**的作用为**管理**所有的**业务对象**，解决对象间的**依赖**，记住是所有的业务对象。包括**数据层**以及**逻辑层**的对象，也可以管理上一节我们讲到的 Hibernate 对象，所有的对象都可以交给 **Spring** 管理。如果项目使用了多个框架，比如**在线商城系统**使用了

(**Hibernate+Spring+SpringMVC**)，Spring 框架可以帮忙整合框架。

最直观地理解，Spring 是一个**容器**，管理业务对象的生命周期，解决对象间的依赖。最后通过 Spring 提供的接口，获取**容器**中的**业务对象**。

## 二、 Spring 特点

### 1. 轻量

从大小与开销两方面而言 Spring 都是**轻量**的。完整的 Spring 框架可以在一个大小只有 1MB 多的 JAR 文件里发布。并且 Spring 所需的处理开销也是微不足道的。此外，Spring 是非侵入式的：典型地，Spring 应用中的对象不依赖于 Spring 的特定类。

### 2. 控制反转

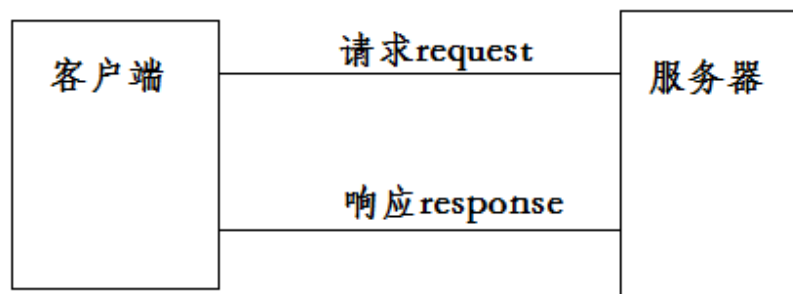
Spring 通过一种称作**控制反转 (IoC)**的技术促进了低耦合。当应用了 IoC，一个对象依赖的其它对象会通过被动的方式传递进来，而不是这个对象自己创建或者查找依赖对象。不是对象从容器中查找依赖，而是容器在对象初始化时不等对象请求就主动将依赖传递给它。

比如有 A 对象和 B 对象，B 对象是 A 对象的一个属性，A、B 对象都交由 Spring 容器管理，当我们需要实例化 A 对象的时候，容器会主动地将 B 对象实例化并注入进 A 对象。解决 A 对象对 B 对象的依赖。

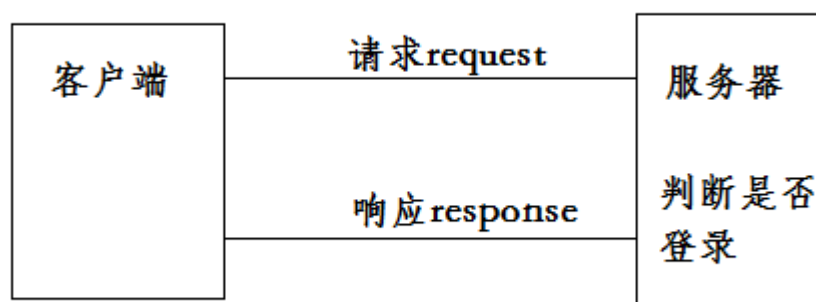
### 3. 面向切面

面向切面的功能，主要涉及到 Spring 的过滤器 (Filter) 和拦截器 (Interceptor)，这里涉及到前面说的服务和响应。

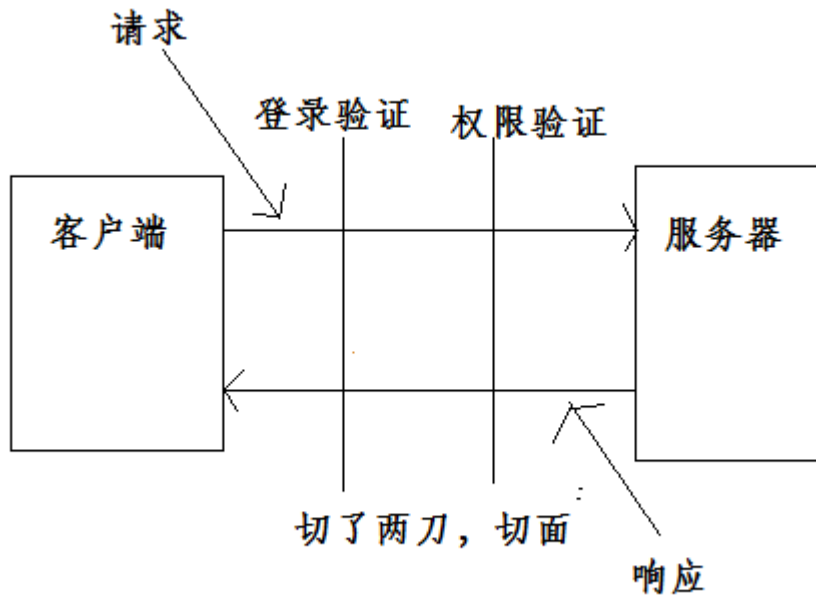
正常情况下，客户端和服务器的请求交互如下：



现在有这么个业务场景：客户端的有些请求，必须要登录后才能够得到处理，如果没登录就不能够进行响应。通常的做法是，在服务器响应的那端判断是否登录，然后做出处理：



然后在 Spring 中，提供了面向切面的处理，所谓面向切面，就是指可以进行请求进行拦截，相当于一刀切进去一样：



切面最常用的就是验证（登录、权限），在请求得到处理之前，对请求进行拦截，判断请求是否登录，判断是否拥有权限，根据验证的结果进行处理还是跳转到登录页面。

#### 4. 容器

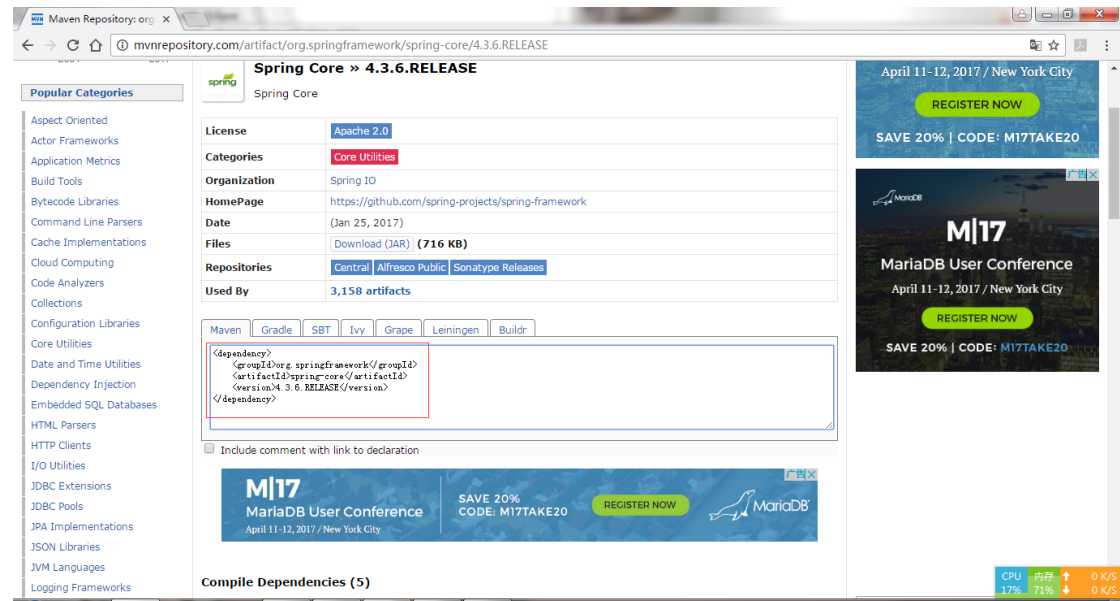
Spring 包含并管理应用对象的配置和生命周期，在这个意义上它是一种容器，可以配置你的每个 bean 如何被创建。

#### 5. MVC

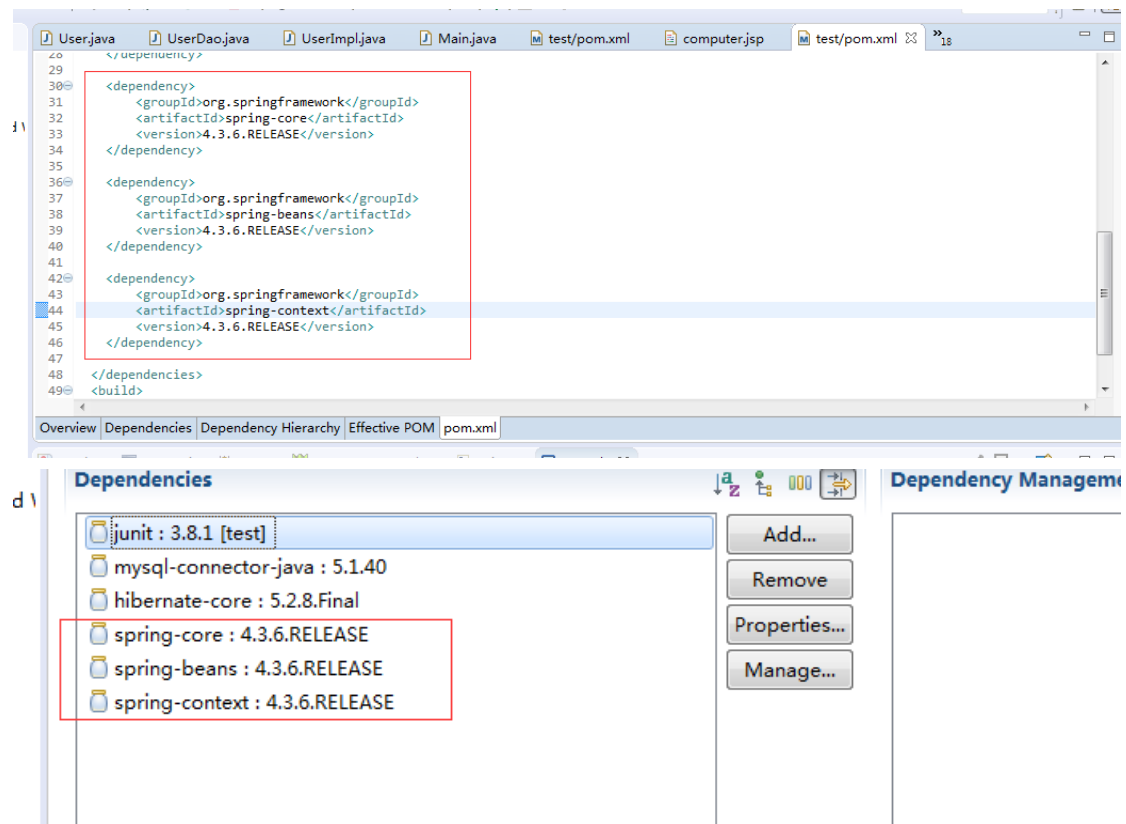
SpringMVC 是 Spring 提供的对 MVC 模式的一种实现，后续有详细讲解。

### 三、Spring 配置

在 Maven 仓库找到 Spring 的依赖：



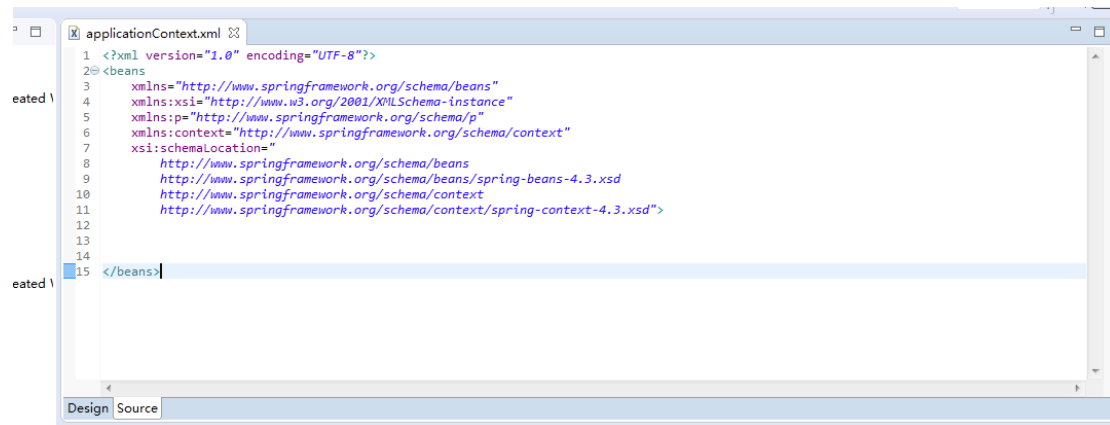
将 Spring 的依赖拷贝到 test 项目的 pom.xml 文件中：



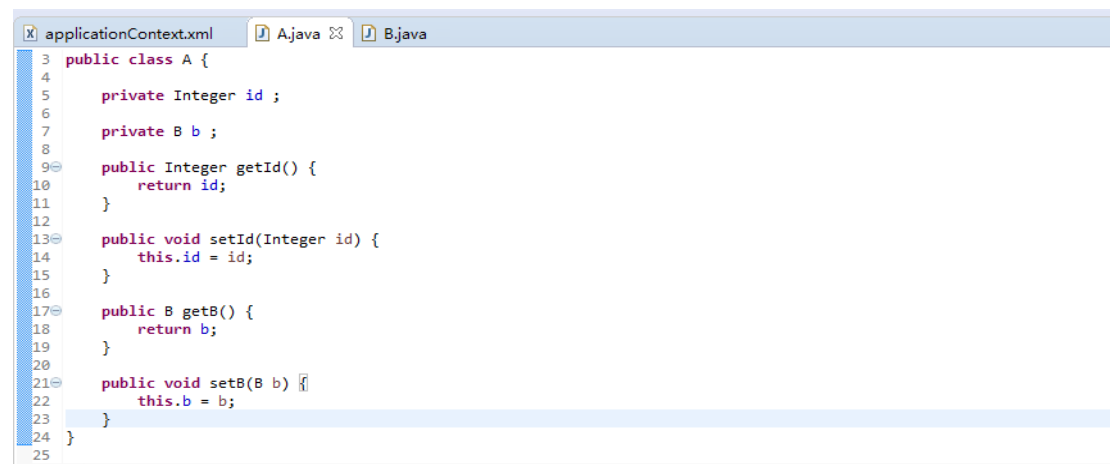
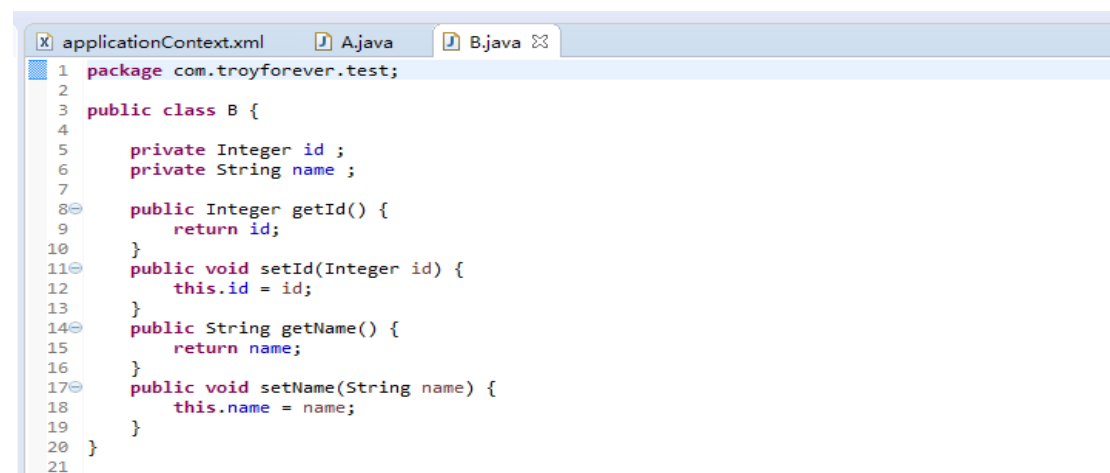
## 四、Spring 简单使用

在 Java Resources -> src/main/resources 新建

applicationContext.xml 文件：

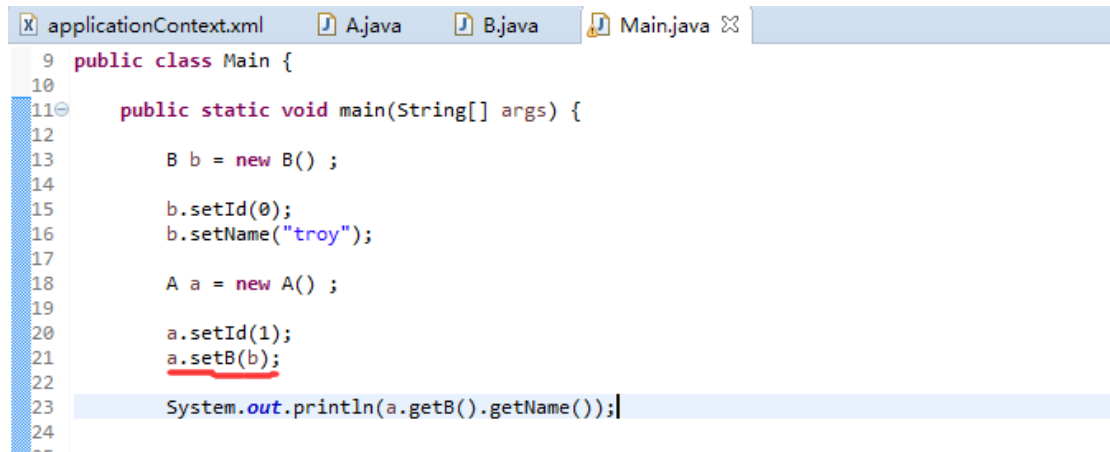


在 com.troyforever.test 包下，新建 A、B 两个类：



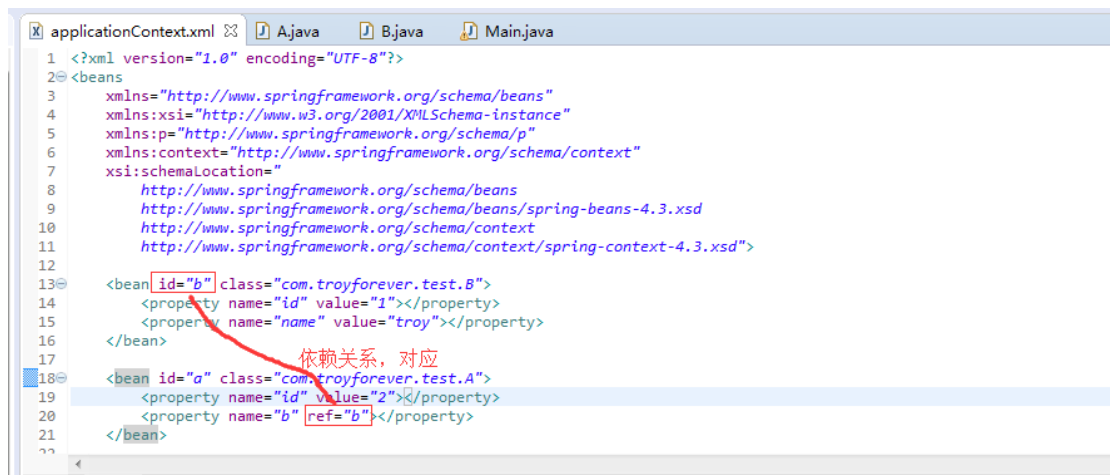


可以看到 B 类对象是 A 类的成员，即 A 类依赖 B 类，正常情况下，想要实例化 A 类对象，首先得实例化 B 类对象，然后将 B 类对象赋给 A 类对象的 b 成员：



```
9 public class Main {
10
11     public static void main(String[] args) {
12
13         B b = new B() ;
14
15         b.setId(0);
16         b.setName("troy");
17
18         A a = new A() ;
19
20         a.setId(1);
21         a.setB(b);
22
23         System.out.println(a.getB().getName());
24     }
```

但是，Spring 容器可以帮助我们解决这个依赖的关系，将对象交给 Spring 容器管理的时候，在 applicationContext 里面配置：



```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans
3     xmlns="http://www.springframework.org/schema/beans"
4     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5     xmlns:p="http://www.springframework.org/schema/p"
6     xmlns:context="http://www.springframework.org/schema/context"
7     xsi:schemaLocation="
8         http://www.springframework.org/schema/beans
9         http://www.springframework.org/schema/beans/spring-beans-4.3.xsd
10        http://www.springframework.org/schema/context
11        http://www.springframework.org/schema/context/spring-context-4.3.xsd">
12
13     <bean id="b" class="com.troyforever.test.B">
14         <property name="id" value="1"></property>
15         <property name="name" value="troy"></property>
16     </bean>
17
18     <bean id="a" class="com.troyforever.test.A">
19         <property name="id" value="2"></property>
20         <property name="b" ref="b"></property>
21     </bean>
22 </beans>
```

下面在 Main 函数里面获取到 B 的实例：

```
12 public class Main {
13
14     public static void main(String[] args) {
15
16         //实例化Spring容器
17         ApplicationContext app = new ClassPathXmlApplicationContext("applicationContext.xml") ;
18
19         //获取B实例
20         B b = (B) app.getBean("b") ;
21
22         System.out.println(b.getName());
23     }
24 }
```

Markers Properties Servers Data Source Explorer Snippets Console Progress

<terminated> Main [Java Application] C:\Java\jdk1.8.0\_121\bin\javaw.exe (2017年3月3日 下午12:04:21)

三月 03, 2017 12:04:21 下午 org.springframework.context.support.ClassPathXmlApplicationContext prepareRefresh: Refreshing org.springframework.context.support.ClassPathXmlApplicationContext@5cb0d902: startup date 三月 03, 2017 12:04:21 下午 org.springframework.beans.factory.xml.XmlBeanDefinitionReader loadBeanDefinitions Loading XML bean definitions from class path resource [applicationContext.xml]

troy

接下来获取 A 的实例：

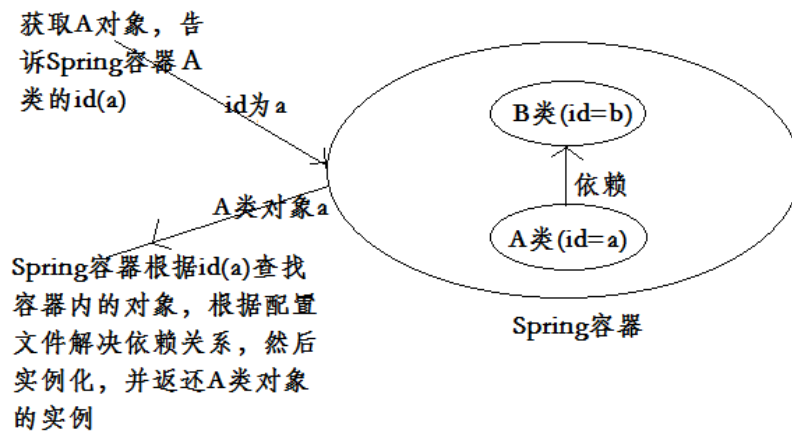
```
12 public class Main {
13
14     public static void main(String[] args) {
15
16         //实例化Spring容器
17         ApplicationContext app = new ClassPathXmlApplicationContext("applicationContext.xml") ;
18
19         //获取A实例
20         A a = (A) app.getBean("a") ;
21
22         System.out.println(a.getB().getName());
23     }
24 }
```

Markers Properties Servers Data Source Explorer Snippets Console Progress

<terminated> Main [Java Application] C:\Java\jdk1.8.0\_121\bin\javaw.exe (2017年3月3日 下午12:05:20)

三月 03, 2017 12:05:20 下午 org.springframework.context.support.ClassPathXmlApplicationContext prepareRefresh: Refreshing org.springframework.context.support.ClassPathXmlApplicationContext@5cb0d902: startup date [Fri Mar 三月 03, 2017 12:05:20 下午 org.springframework.beans.factory.xml.XmlBeanDefinitionReader loadBeanDefinitions Loading XML bean definitions from class path resource [applicationContext.xml]

troy



这里我们可以看到，我们不用手动去实例 B 的对象然后赋值给 A，在 Spring 容器里面，它帮我们解决好了依赖关系，我们只需要告诉容器我们需要哪个对象（通过 id 属性标识），它就将解决好了依赖关系的对象实例化给我们。