一：Spring Bean概念

由 Spring IoC 容器管理的对象称为 Bean，Bean 根据 Spring 配置文件中的信息创建。

我们可以把 Spring IoC 容器看作是一个大工厂，Bean 相当于工厂的产品。如果希望这个大工厂生产和管理 Bean，就需要告诉容器需要哪些 Bean，以哪种方式装配。

Spring 配置文件支持两种格式，即 XML 文件格式和 Properties 文件格式。

Properties 配置文件主要以 key-value 键值对的形式存在，只能赋值，不能进行其他操作，适用于简单的属性配置。

XML 配置文件采用树形结构，结构清晰，相较于 Properties 文件更加灵活。但是 XML 配置比较繁琐，适用于大型的复杂的项目。

通常情况下，Spring 的配置文件都是使用 XML 格式的。XML 配置文件的根元素是 <beans>，该元素包含了多个子元素 <bean>。每一个 <bean> 元素都定义了一个 Bean，并描述了该 Bean 是如何被装配到 Spring 容器中的。

例如，在《第一个Spring程序》一节中的 Beans.xml 配置文件，代码如下所示：

```
<?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-3.0.xsd">    <bean
id="helloWorld" class="net.biancheng.c.HelloWorld">        <property name="message"
value="Hello World!"/>    </bean></beans>
```

在 XML 配置的<beans> 元素中可以包含多个属性或子元素，常用的属性或子元素如下表所示。

属性名称	描述
id	Bean 的唯一标识符，Spring IoC 容器对 Bean 的配置和管理都通过该属性完成。id 的值必须以字母开始，可以使用字母、数字、下划线等符号。
name	该属性表示 Bean 的名称，我们可以通过 name 属性为同一个 Bean 同时指定多个名称，每个名称之间用逗号或分号隔开。Spring 容器可以通过 name 属性配置和管理容器中的 Bean。
class	该属性指定了 Bean 的具体实现类，它必须是一个完整的类名，即类的全限定名。
scope	表示 Bean 的作用域，属性值可以为 singleton（单例）、prototype（原型）、request、session 和 global Session。默认值是 singleton。
constructor-arg	<bean> 元素的子元素，我们可以通过该元素，将构造参数传入，以实现 Bean 的实例化。该元素的 index 属性指定构造参数的序号（从 0 开始），type 属性指定构造参数的类型。
property	<bean>元素的子元素，用于调用 Bean 实例中的 setter 方法对属性进行赋值，从而完成属性的注入。该元素的 name 属性用于指定 Bean 实例中相应的属性名。
ref	<property> 和 <constructor-arg> 等元素的子元素，用于指定对某个 Bean 实例的引用，即 <bean> 元素中的 id 或 name 属性。
value	<property> 和 <constractor-arg> 等元素的子元素，用于直接指定一个常量值。
list	用于封装 List 或数组类型的属性注入。
set	用于封装 Set 类型的属性注入。
map	用于封装 Map 类型的属性注入。
entry	<map> 元素的子元素，用于设置一个键值对。其 key 属性指定字符串类型的键值，ref 或 value 子元素指定其值。
init-method	容器加载 Bean 时调用该方法，类似于 Servlet 中的 init() 方法
destroy-method	容器删除 Bean 时调用该方法，类似于 Servlet 中的 destroy() 方法。该方法只在 scope=singleton 时有效
lazy-init	懒加载，值为 true，容器在首次请求时才会创建 Bean 实例；值为 false，容器在启动时创建 Bean 实例。该方法只在 scope=singleton 时有效

/

二 Spring Bean属性注入

1.构造函数注入，主要是依赖constructor-arg属性

2.set注入，主要是依赖property

三 Spring 自动装配

Spring 的自动装配功能可以让 Spring 容器依据某种规则（自动装配的规则，有五种），为指定的 Bean 从应用的上下文（AppplicationContext 容器）中查找它所依赖的 Bean，并自动建立 Bean 之间的依赖关系。而这一过程是在完全不使用任何 <constructor-arg>和 <property> 元素 ref 属性的情况下进行的。

Spring 的自动装配功能能够有效地简化 Spring 应用的 XML 配置，因此在配置数量相当多时采用自动装配降低工作量。

Spring 框架式默认不支持自动装配的，要想使用自动装配，则需要对 Spring XML 配置文件中 <bean> 元素的 autowire 属性进行设置。/

```
<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-3.0.xsd">

    <!--部门 Dept 的 Bean 定义-->
    <bean id="dept" class="net.biancheng.c.Dept"></bean>

    <!--雇员 Employee 的 Bean 定义,通过 autowire 属性设置自动装配的规则-->
    <bean id="employee" class="net.biancheng.c.Employee" autowire="byName">
    </bean>
</beans>
```

自动装配规则Spring 共提供了 5 中自动装配规则，它们分别与 autowire 属性的 5 个取值对应，具体说明如下表。

属性值	说明
byName	按名称自动装配。 Spring 会根据的 Java 类中对象属性的名称，在整个应用的上下文 ApplicationContext（IoC 容器）中查找。若某个 Bean 的 id 或 name 属性值与这个对象属性的名称相同，则获取这个 Bean，并与当前的 Java 类 Bean 建立关联关系。
byType	按类型自动装配。 Spring 会根据 Java 类中的对象属性的类型，在整个应用的上下文 ApplicationContext（IoC 容器）中查找。若某个 Bean 的 class 属性值与这个对象属性的类型相匹配，则获取这个 Bean，并与当前的 Java 类的 Bean 建立关联关系。
constructor	与 byType 模式相似，不同之处在与它应用于构造器参数（依赖项），如果在容器中没有找到与构造器参数类型一致的 Bean，那么将抛出异常。 其实就是根据构造器参数的数据类型，进行 byType 模式的自动装配。
default	表示默认采用上一级元素 <beans> 设置的自动装配规则（default-autowire）进行装配。
no	默认值，表示不使用自动装配，Bean 的依赖关系必须通过 <constructor-arg>和 <property> 元素的 ref 属性来定义。

基于注解的自动装配

从 Java 5 开始，Java 增加了对注解（Annotation）的支持，它是代码中的一种特殊标记，可以在编译、类加载和运行时被读取，执行相应的处理。开发人员可以通过注解在不改变原有代码和逻辑的情况下，在源代码中嵌入补充信息。

Spring 从 2.5 版本开始提供了对注解技术的全面支持，我们可以使用注解来实现自动装配，简化 Spring 的 XML 配置。

Spring 通过注解实现自动装配的步骤如下：

引入依赖

开启组件扫描

使用注解定义 Bean

依赖注入

1. 引入依赖使用注解的第一步，就是要在项目中引入以下 Jar 包。

org.springframework.core-5.3.13.jar

org.springframework.beans-5.3.13.jar

spring-context-5.3.13.jar

spring-expression-5.3.13.jar

commons.logging-1.2.jar

spring-aop-5.3.13.jar

注意，除了 spring 的四个基础 jar 包和 commons-logging-xxx.jar 外，想要使用注解实现 Spring 自动装配，还需要引入Spring 提供的 spring-aop 的 Jar 包。

2. 开启组件扫描Spring 默认不使用注解装配 Bean，因此我们需要在 Spring 的 XML 配置中，通过 <context:component-scan> 元素开启 Spring Beans的自动扫描功能。开启此功能后，Spring 会自动从扫描指定的包（base-package 属性设置）及其子包下的所有类，如果类上使用了 @Component 注解，就将该类装配到容器中。

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
    http://www.springframework.org/schema/context
            http://www.springframework.org/schema/context/spring-context.xsd">
    <!--开启组件扫描功能-->
    <context:component-scan base-package="net.biancheng.c"></context:component-scan>
</beans>
```

注意：在使用 <context:component-scan> 元素开启自动扫描功能前，首先需要在 XML 配置的一级标签 <beans> 中添加 context 相关的约束

```
32
```

3. 使用注解定义 Bean

Spring 提供了以下多个注解，这些注解可以直接标注在 Java 类上，将它们定义成 Spring Bean。

注解	说明
@Component	该注解用于描述 Spring 中的 Bean，它是一个泛化的概念，仅仅表示容器中的一个组件（Bean），并且可以作用在应用的任何层次，例如 Service 层、Dao 层等。 使用时只需将该注解标注在相应类上即可。
@Repository	该注解用于将数据访问层（Dao 层）的类标识为 Spring 中的 Bean，其功能与 @Component 相同。
@Service	该注解通常作用在业务层（Service 层），用于将业务层的类标识为 Spring 中的 Bean，其功能与 @Component 相同。
@Controller	该注解通常作用在控制层（如 Struts2 的 Action、SpringMVC 的 Controller），用于将控制层的类标识为 Spring 中的 Bean，其功能与 @Component 相同。

这个地方有一点要注意直接使用@Component注解和直接使用明确的注解比如@Repository，@Service等区别在于使用特定的注解相当于继承了@Component的基本功能，然后添加了属于自己的特定功能，比如说@Repository还提供了数据库操作相关的事务管理，数据持久化相关的对应功能等。

```
Spring框架提供了多种基于注解的方式来实现依赖注入。以下是一些常用的注解实现依赖注入的方式：

1. `@Autowired`注解：
    - `@Autowired`注解用于自动装配依赖对象，通过类型匹配来查找并注入依赖。
    - 可以将`@Autowired`注解应用于构造方法、成员变量、Setter方法或普通方法上。

2. `@Qualifier`注解：
    - 当存在多个类型匹配的依赖对象时，可以与`@Autowired`注解一起使用，通过指定名称来进行注入。
    - 在使用`@Qualifier`注解时，通常结合`@Autowired`注解一起使用，例如：`@Autowired @Qualifier("beanName")`。

3. `@Resource`注解：
    - `@Resource`注解是Java EE提供的注解，也可以用于实现依赖注入。
    - 它可以根据名称或类型进行依赖注入，类似于`@Autowired`和`@Qualifier`的组合。
    - 通常用于注入其他组件、数据库资源、JNDI资源等。

4. `@Value`注解：
    - `@Value`注解用于注入简单类型的值或表达式，如基本数据类型、字符串、SpEL表达式等。
    - 可以将`@Value`注解应用于成员变量、Setter方法或普通方法上。

5. `@Inject`注解：
    - `@Inject`注解是JSR-330中定义的注解，类似于`@Autowired`注解，也用于实现依赖注入。
    - 可以将`@Inject`注解应用于构造方法、成员变量、Setter方法或普通方法上。
    - 需要导入javax.inject依赖，例如：`import javax.inject.Inject;`。

```

这些注解提供了不同的依赖注入方式，可以根据具体的需求和场景选择适合的注解。它们使得依赖对象的注入更加简洁和方便，减少了手动配置和编写繁琐的依赖关系。

*** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** ***
*** *** *** *** *** *** *** *** *** *** *** *** ***

这里有一点需要重点说明一下，那个地方需要把其他的类注入到自身中，就在引用的对应类的方法或者属性上面写相应的注解例如@Autowired或者@Resource

也就是说@Componet注解相当于把对应的类写道了beans.xml中了，@Autowired相当于明确需要注入的且在beans.xml中的其他类

*** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** ***
*** *** *** *** *** *** *** *** *** *** *** *** ***