

Spring的Bean管理（XML方式）

- 三种实例化Bean的方式
 - 使用类构造器实例化(默认无参数)
 - 使用静态工厂方法实例化(简单工厂模式)
 - 使用实例工厂方法实例化(工厂方法模式)

Bean的配置

- id和name
 - 一般情况下，装配一个Bean时，通过指定一个id属性作为Bean的名称
 - id属性在IOC容器中必须是唯一的
 - 如果Bean的名称中含有特殊字符，就必须使用name属性
 - id不能带有特殊字符
- class
 - class用于设置一个类的完全路径名称，主要作用是IOC容器生成类的实例
- scope属性

类别	说明
singleton	在SpringIOC容器中仅存在一个Bean实例，Bean以单实例的方式存在
prototype	每次调用getBean()时都会返回一个新的实例
request	每次HTTP请求都会创建一个新的Bean，该作用域仅适用于WebApplicationContext环境
session	同一个HTTP Session共享一个Bean，不同的HTTP Session使用不同的Bean。该作用域仅适用于WebApplicationContext环境

scope属性

Spring容器中Bean的生命周期

Spring初始化bean或销毁bean时，有时需要做一些处理工作，因此spring可以在创建和销毁bean的时候调用bean的两个生命周期方法。

- 当bean被载入到容器的时候调用init
- 当bean从容器中删除的时候调用destroy(scope=singleton有效)
- 完整的生命周期
 1. instantiate bean对象实例化
 2. populate properties 封装属性
 3. 如果Bean实现BeanNameAware 执行setBeanName
 4. 如果Bean实现BeanFactoryAware 或者ApplicationContextAware 设置工厂setBeanFactroy 或者上下文对象setApplicationContext

5. 如果存在类实现BeanPostProcessor(后处理Bean), 执行postProcessBeforeInitialization
6. 如果Bean实现InitializingBean执行afterPropertiesSet
7. 调用 `<bean init-method="init">` 指定初始化方法init
8. 如果存在类实现BeanPostProcessor(处理Bean), 执行postProcessAfterInitialization
9. 执行业务处理
10. 如果Bean实现DisposableBean执行destroy
11. 调用 `<bean destroy-method="customerDestroy">` 指定销毁方法customerDestroy

Spring的属性注入

- 对于类成员变量, 注入方式有三种

- 构造函数注入

- 通过构造方法注入Bean的属性值或依赖的对象, 它保证了Bean实例在实例化后就可以使用
- 构造器注入在 `<constructor-arg>` 元素里声明的属性

```
1 <bean id="user" class="com.hn.ioc.demo4.User">
2     <constructor-arg name="name" value="张三"/>
3     <constructor-arg name="age" value="23"/>
4 </bean>
```

- 属性setter方法注入

- 使用set方法注入, 在Spring配置文件中, 通过 `<property>` 设置注入的属性

- 接口注入

- Spring支持前两种

- p名称空间

- 使用p命名空间
- 为了简化XML文件配置, Spring从2.5开始引入一个新的p名称空间
- `p:<属性名>="xxx"` 引入常量值
- `p:<属性名>-ref="xxx"` 引用其它Bean对象

```
1 <beans xmlns="http://www.springframework.org/schema/beans"
2     xmlns:p="http://www.springframework.org/schema/p"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xsi:schemaLocation="http://www.springframework.org/schema/beans
5 http://www.springframework.org/schema/beans/spring-beans.xsd">
6     <bean id="person" class="com.hn.ioc.demo4.Person" p:name="大黄" p:age="50" p:cat-
7 ref="cat"/>
8
9     <bean id="cat" class="com.hn.ioc.demo4.Cat" p:name="Tomcat233"/>
```

- SpEL注入

- SpEL:spring expression language , spring表达式语言, 对依赖注入进行简化
- 语法: `#{表达式}`
- `<bean id="" value="#{表达式}">`

- SpEL表达式语言

- 语法: `{}`
- `{ 'hello' }`: 使用字符串
- `{beanId}`: 使用另一个bean
- `{beanId.content.toUpperCase() }`: 使用指定名属性, 并使用方法
- `{T(java.lang.Math).PI}`: 使用静态字段或方法

- ```
1 <bean id="category" class="com.hn.ioc.demo4.Category">
2 <property name="name" value="{ '服装' }"/>
3 </bean>
4
5 <bean id="produceInfo" class="com.hn.ioc.demo4.ProdictInfo"/>
6
7 <bean id="product" class="com.hn.ioc.demo4.Product">
8 <property name="name" value="{ '男装' }"/>
9 <property name="price" value="{produceInfo.calculatePrice() }"/>
10 <property name="category" value="{category}"/>
11 </bean>
```

1

- 复杂类型的属性注入

- 数组类型的属性注入

- ```
1 <property name="ars">
2     <list>
3         <value>aaa</value>
4         <value>bbb</value>
5         <value>ccc</value>
6         <value>ddd</value>
7     </list>
8 </property>
```

- List集合类型的属性注入

- ```
1 <property name="list">
2 <list>
3 <value>11</value>
4 <value>22</value>
5 <value>33</value>
6 </list>
7 </property>
```

- Set集合类型的属性注入

```

1 <property name="set">
2 <set>
3 <value>ddd</value>
4 <value>eee</value>
5 <value>fff</value>
6 </set>
7 </property>

```

- Map集合类型的属性注入

```

1 <property name="map">
2 <map>
3 <entry key="a" value="111"></entry>
4 <entry key="b" value="222"></entry>
5 <entry key="c" value="333"></entry>
6 </map>
7 </property>

```

- Properties类型的属性注入

```

1 <property name="properties">
2 <props>
3 <prop key="username">root</prop>
4 <prop key="password">123456</prop>
5 </props>
6 </property>

```

## Spring的Bean管理(注解方式)

### 使用注解定义Bean

- Spring2.5引入使用注解定义Bean
  - @Component 描述Spring框架中的Bean
- 除了@Component外，Spring提供了3个功能基本和@Component等效的注解
  - @Repository用于对DAO实现类进行标注
  - @Service用于对Service实现类进行标注
  - @Controller用于对Controller实现类进行标注
- 这三个注解是为了让标注类本身的用途清晰，Spring在后续版本会对其增强

### Spring的属性注入---注解方式

- 普通属性值:@Value
- 使用@Autowired进行自动注入
- @Autowired默认按照类型进行注入
  - 如果存在两个相同Bean类型相同，则按照名称注入

- @Autowired注入时可以针对成员变量或者set方法
- 通过@Autowired的required属性，设置一定要找到匹配的Bean
- 通过@Qualifier指定注入Bean的名称
- 使用@Qualifier指定Bean名称后，注解Bean必须指定相同名称
- Spring提供对JSR-250中定义@Resource标准注解的支持
- @Resource和@Autowired注解功能相似

## Spring的其他注解

Spring初始化Bean或销毁Bean时，有时需要作一些处理工作，因此spring可以在创建和拆卸bean的时候调用bean的两个生命周期方法。

```
<bean id= "foo" class= "...Foo"
 init-method= "setup"
 destroy-method= "teardown" />
```

当bean被载入到容器的时候调用  
**setup**  
注解方式下:

- @PostConstruct
- 初始化

当bean从容器中删除的时候调用  
**teardown(scope= singleton有效)**  
注解方式如下:

- @PreDestroy
- 销毁

## Bean的作用范围注解

- 使用注解配置的Bean和 `<bean>` 配置的一样，默认作用范围都是singleton
- @Scope注解用于指定Bean的作用范围

## 传统XML配置和注解配置混合使用

- XML方式的优势
  - 结构清晰，易于阅读
- 注解方式的优势
  - 开发便捷，属性注入方便，无须set方法
- XML与注解的整合开发
  - 1.引入context命名空间
  - 2.在配置文件中添加context:annotation-config标签
  - `<context:annotation-config>`