

# Day04

---

## IoC配置 (XML格式)

- bean

  - 格式

  - 基本属性

- bean 属性 scope

  - 格式

  - 取值

- bean 生命周期

  - 格式

  - 取值

  - 注意事项:

- 延迟初始化

## 课堂练习

- bean对象创建方式

  - 格式:

- 了解 IoC 和 DI

## DI 配置

- set 注入(常用)

  - 格式

  - 基本属性

- 构造器注入

  - 格式

  - 基本属性

  - 其他属性

  - 说明

- 集合类型数据注入

  - 格式

  - list

properties

array

set

map

使用p命名空间简化配置

格式

SpEL

格式

properties 文件

操作步骤

导入

格式

Spring容器加载多个配置文件

bean定义冲突问题

ApplicationContext

BeanFactory

BeanFactory 和 ApplicationContext 对比

FileSystemXmlApplicationContext 和 ClassPathXmlApplicationContext对比

第三方资源配置

Druid

课后作业

## IoC配置（XML格式）

Property	Explained in...
Class	<a href="#">Instantiating Beans</a>
Name	<a href="#">Naming Beans</a>
Scope	<a href="#">Bean Scopes</a>
Lazy initialization mode	<a href="#">Lazy-initialized Beans</a>
Initialization method	<a href="#">Initialization Callbacks</a>
Destruction method	<a href="#">Destruction Callbacks</a>

## bean

bean标签归属于 beans 标签下，作用就是把资源放入 Spring 容器中，让该资源被 Spring 控制和管理。

### 格式

XML | 复制代码

```

1 <beans>
2   <bean />
3 </beans>

```

### 基本属性

XML | 复制代码

```

1 <bean id="beanId" name="beanName1,beanName2" class="ClassName"></bean>

```

#### 属性说明

- id: bean的名称，通过id值获取bean
- class: bean的类型
- name: bean的名称，可以通过name值获取bean，用于多人配合时给bean起别名

## bean 属性 scope

scope 的作用是定义 bean 的作用范围

## 格式

```
1 <bean scope="singleton"></bean>
```

XML

复制代码

## 取值

- singleton: (Default)设定创建出的对象保存在spring容器中，是一个单例的对象
- prototype: 设定创建出的对象保存在spring容器中，是一个非单例的对象
- request、session、application、websocket：设定创建出的对象放置在web容器对应的位置

## bean 生命周期

bean 提供了两个属性声明声明周期

- init-method
- destroy-method

作用是定义 bean 对象在初始化或销毁时完成的工作

## 格式

```
1 <bean init-method="init" destroy-method="destroy"></bean>
```

XML

复制代码

## 取值

bean对应的类中对应的具体方法名

## 注意事项：

- 当scope="singleton"时，spring容器中有且仅有一个对象，init方法在创建容器时仅执行一次
- 当scope="prototype"时，spring容器要创建同一类型的多个对象，init方法在每个对象创建时均执行一次
- 当scope="singleton"时，关闭容器会导致bean实例的销毁，调用destroy方法一次
- 当scope="prototype"时，对象的销毁由垃圾回收机制gc()控制，destroy方法将不会被执行

## 延迟初始化

lazy-init 属性可以让 bean 在获取的时候再初始化，而不是读取spring 配置文件时就初始化。

XML | 复制代码

```
1 <bean id="userService" lazy-init="true" init-method="init" destroy-  
  method="cleanup" class="com.rushuni.service.impl.UserServiceImpl"></bean>
```

## 课堂练习

- 创建 spring 项目，熟悉 bean 的各个配置。

## bean对象创建方式

定义bean对象创建方式，使用静态工厂的形式创建bean，兼容早期遗留系统的升级工作  
使用 factory-method 属性可以定义bean对象创建方式，使用静态工厂的形式创建bean。

格式：

XML | 复制代码

```
1 <bean class="FactoryClassName" factory-method="factoryMethodName"></bean>
```

factory-method：工厂 bean 中用于获取对象的静态方法名

注意：class属性必须配置成静态工厂的类名

通过 factory-bean 属性，使用实例工厂的形式创建bean，兼容早期遗留系统的升级工作

XML | 复制代码

```
1 <bean factory-bean="factoryBeanId" factory-method="factoryMethodName"></bean>
```

factory-bean：实例工厂的beanId，工厂bean中用于获取对象的实例方法名

注意：使用实例工厂创建bean首先需要将实例工厂配置bean，交由spring进行管理。

## 了解 IoC 和 DI

- IoC (Inversion Of Control) 控制翻转，Spring反向控制应用程序所需要使用的外部资源
- DI (Dependency Injection) 依赖注入，应用程序运行依赖的资源由Spring为其提供，资源

进入应用程序的方式称为注入

IoC 与 DI 是同一件事站在不同角度看待问题：

- IoC 强调等待 Spring 容器提供资源。
- DI 是强调主动权在 Spring 手中。

## DI 配置

### set 注入(常用)

通过 property 子标签，Spring 容器将使用 set 方法的形式为 bean 提供资源。

#### 格式

```
1 <bean>
2   <property />
3 </bean>
```

XML

复制代码

#### 基本属性

```
1 <property name="propertyName" value="propertyValue" ref="beanId"/>
```

XML

复制代码

说明：

- name：对应bean中的属性名，要求该属性必须提供可访问的set方法（严格规范为此名称是set方法对应名称）。
- value：设定非引用类型属性对应的值，不能与ref同时使用。
- ref：设定引用类型属性对应bean的id，不能与value同时使用。

一个bean可以有多个property标签。

### 构造器注入

通过 constructor-arg 子标签，可以让 Spring 容器使用构造方法的形式为bean提供资源。

#### 格式

```

1 <bean>
2   <constructor-arg />
3 </bean>

```

## 基本属性

```

1 <constructor-arg name="argsName" value="argsValue" />

```

说明：

- name：对应bean中的构造方法所携带的参数名
- value：设定非引用类型构造方法参数对应的值，不能与ref同时使用

## 其他属性

```

1 <constructor-arg index="arg-index" type="arg-type" ref="beanId"/>

```

## 说明

- ref：设定引用类型构造方法参数对应bean的id，不能与value同时使用
- type：设定构造方法参数的类型，用于按类型匹配参数或进行类型校验
- index：设定构造方法参数的位置，用于按位置匹配参数，参数index值从0开始计数

一个 bean 可以有多个 constructor-arg 标签

## 集合类型数据注入

在 property 标签或者 constructor-arg 标签中，可以使用 array, list, set, map, properties 这些子标签来注入集合数据类型的属性。

## 格式


```

1 <property>
2   <list></list>
3 </property>

```

## list


XML

 复制代码

```
1 <property name="mylist">
2   <list>
3     <value>rushuni</value>
4     <value>123</value>
5   </list>
6 </property>
```

## properties


XML

 复制代码

```
1 <property name="myproperties">
2   <props>
3     <prop key="name">rushuni</prop>
4     <prop key="value">123</prop>
5   </props>
6 </property>
```

## array

XML

 复制代码

```
1 <property name="myarray">
2   <array>
3     <value>123456</value>
4     <value>66666</value>
5   </array>
6 </property>
```

## set



XML | 复制代码

```
1 <property name="myset">
2   <set>
3     <value>rushuni</value>
4     <value>123</value>
5   </set>
6 </property>
```

## map

XML | 复制代码

```
1 <property name="mymap">
2   <map>
3     <entry key="name" value="rushuni"/>
4     <entry key="value" value="123"/>
5   </map>
6 </property>
```

## 使用p命名空间简化配置

通过 p:propertyName, p:propertyName-ref 这两个属性，也可以为bean注入属性值

## 格式

XML | 复制代码

```
1 <bean p:propertyName="propertyValue" p:propertyName-ref="beanId"/>
```

注意：使用p命名空间需要先开启 spring 对 p 命名空间的支持，在 beans 标签中添加对应空间支持

XML | 复制代码

```
1 <beans xmlns="http://www.springframework.org/schema/beans"
2       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3       xmlns:p="http://www.springframework.org/schema/p"
4       xsi:schemaLocation="http://www.springframework.org/schema/beans
5       http://www.springframework.org/schema/beans/spring-beans.xsd">
```

xml 文件中开启命名空间的方式就是在 beans 开始标签中添加。

例子：

```
1 <bean
2     id="userService"
3     class="com.rushuni.service.impl.UserServiceImpl"
4     p:userDao-ref="userDao"
5 />
```

XML | 复制代码

## SpEL

通过 value 属性，Spring提供了 EL 表达式，也可以为 bean 注入属性值。

### 格式

```
1 <property value="EL"></bean>
```

XML | 复制代码

- 注意：所有属性值不区分是否引用类型，统一使用value赋值
- 所有格式统一使用 value="\*\*\*\*\*"
- 常量 #{1} #{3.14} #{2e5} #{'apple'}
- 引用bean #{beanId}
- 引用bean属性 #{beanId.propertyName}
- 引用bean方法 beanId.methodName().method2()
- 引用静态方法 T(java.lang.Math).PI
- 运算符支持 #{3 lt 4 == 4 ge 3}
- 正则表达式支持 #{user.name matches '[a-z]{6,}'}

- 集合支持 `#{likes[3]}`

例子：

```
XML | 复制代码
1 <bean id="userService" class="com.rushuni.service.impl.UserServiceImpl">
2   <property name="userDao" value="#{userDao}"/>
3   <property name="age" value="#{22}"/>
4 </bean>
```

## properties 文件

Spring提供了读取外部properties文件的机制，使用读取到的数据为bean的属性赋值

### 操作步骤

1. 准备外部properties文件
2. 开启context命名空间支持

```
XML | 复制代码
1 xmlns:context="http://www.springframework.org/schema/context"
```

3. 加载指定的properties文件

```
XML | 复制代码
1 <context:property-placeholder location="classpath:filename.properties">
```

4. 使用加载的数据

```
XML | 复制代码
1 <property name="propertyName" value="${propertiesName}"/>
```

如果需要加载所有的properties文件，可以使用`\*.properties`表示加载所有的properties文件

读取数据使用\*\*\${propertiesName}\*\*格式进行，其中 propertiesName 指properties文件中的属性名

## 导入

通过 import 标签，可以在当前配置文件中导入其他配置文件中的项

### 格式

```
1 <beans>
2   <import />
3 </beans>
```

XML

复制代码

例子：

```
1 <import resource="config.xml"/>
```

XML

复制代码

说明：

resource：加载的配置文件名

## Spring容器加载多个配置文件

```
1 new ClassPathXmlApplicationContext("config1.xml","config2.xml");
```

Java

复制代码

## bean定义冲突问题

- 同id的bean，后定义的覆盖先定义的
- 导入配置文件可以理解为将导入的配置文件复制粘贴到对应位置
- 导入配置文件的顺序与位置不同可能会导致最终程序运行结果不同

## ApplicationContext

1. ApplicationContext是一个接口，提供了访问spring容器的API
2. ClassPathXmlApplicationContext是一个类，实现了上述功能
3. ApplicationContext的顶层接口是BeanFactory
4. BeanFactory定义了bean相关的最基本操作
5. ApplicationContext在BeanFactory基础上追加了若干新功能

## BeanFactory

```
1 Resource res = new ClassPathResource("applicationContext.xml");
2 BeanFactory bf = new XmlBeanFactory(res);
3 UserService userService = (UserService)bf.getBean("userService");
```

Java

复制代码

## BeanFactory 和 ApplicationContext 对比

- BeanFactory创建的bean采用延迟加载形式，使用才创建
- ApplicationContext创建的bean默认采用立即加载形式

## FileSystemXmlApplicationContext 和 ClassPathXmlApplicationContext对比

- FileSystemXmlApplicationContext可以加载文件系统中任意位置的配置文件
- ClassPathXmlApplicationContext 只能加载类路径下的配置文件

## 第三方资源配置

### Druid

```
1 <bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource">
2   <property name="driverClassName" value="com.mysql.jdbc.Driver"></property>
3   <property name="url" value="jdbc:mysql://localhost:3306/spring_ioc">
4     </property>
5     <property name="username" value="root"></property>
6     <property name="password" value="root"></property>
7   </bean>
```

## 课后作业

- 过一遍今天所有内容，自己完成 IoC 和 DI 的相关例子。
- 复习 MyBatis 的内容。