极客学院
jikexueyuan.com

Spring 配置文件浅析

Spring配置文件浅析 — **课程概要**

- Spring 的配置文件概述
- Spring Bean 的命名
- Spring Bean 的实例化
- Spring Bean 的作用域
- 配置文件的整合

Spring 的配置文件概述

Spring 的配置文件概述

Spring 的配置文件是用于指导 Spring 工厂进行 Bean 的生产、依赖关系注入及 Bean 实例分发的“图纸”，它是一个或多个标准的XML文档，J2EE 程序员必须学会并灵活应用这份“图纸”，准确地表达自己的“生产意图”：

- Spring 配置文件示例
- Spring 容器高层视图
- 基于 XML 的配置

Spring 的配置文件概述 – Spring配置文件示例

Spring 配置文件的一般结构如下：

- **<beans>**
- **<import resource=“resource1.xml” />**
- **<import resource=“resource2.xml” />**
- **<bean id=“bean1” class=“***” ></bean>**
- **<bean name=“bean2” class=“***” ></bean>**
- **<alias alias=“bean3” name=“bean2” />**
- **</beans>**

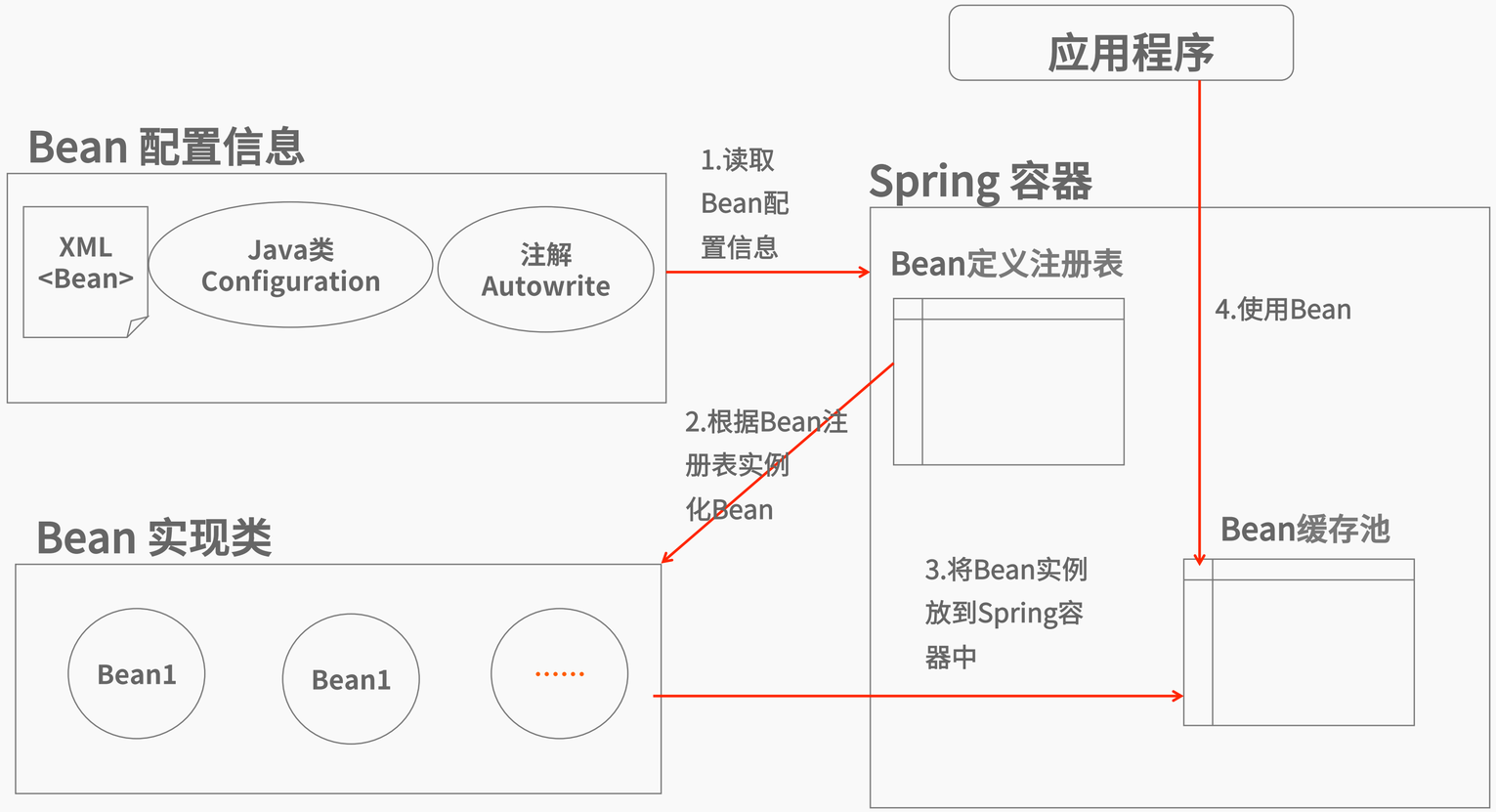
Spring 的配置文件概述 – Spring 容器高层视图

Spring容器启动基本条件：

- Spring 的框架类包
- Bean 的配置信息
- Bean 的实现类

Bean 的元数据信息：

- Bean 的实现类
- Bean 的属性信息
- Bean 的依赖关系
- Bean 的行为配置
- Bean 的创建方式



Spring 的配置文件概述 – 基于 XML 的配置

Spring 的配置文件是基于XML格式的，Spring1.0的配置文件采用DTD格式，Spring2.0以后使用Schema的格式，后者让不同类型的配置拥有了自己的命名空间，是配置文件更具有扩展性。

采取基于Schema的配置格式，文件头的声明会复杂一些，请看一个简单示例：

Spring 的配置文件概述 – 基于 XML 的配置

Spring3.0 的配置Schema文件分布在各模块类包中，如果模块拥有对应的Schema文件，则可以在模块类包中找到一个config目录，Schema文件就位于该目录中，如下是对这些Schema文件的用途进行了简单说明：

- 示例说明：**Spring-beans-3.0.xsd**
- 命名空间：**http://www.springframework.org/schema/beans**
- Schema 文件：**http://www.springframework.org/schema/beans/spring-beans-3.0.xsd**

- **Spring-beans-3.0.xsd**
- **Spring-aop-3.0.xsd**
- **Spring-tx-3.0.xsd**
- **Spring-mvc-3.0.xsd**
- **Spring-util-3.0.xsd**
- **Spring-jee-3.0.xsd**
- **Spring-jdbc-3.0.xsd**
- **Spring-jms-3.0.xsd**
- **Spring-lang-3.0.xsd**
- **Spring-oxm-3.0.xsd**
- **Spring-task-3.0.xsd**
- **Spring-tool-3.0.xsd**

Spring Bean 的命名

Spring Bean 的命名

每个Bean可以有一个或多个 id，我们把第一个 id 称为“标识符”，其余id叫做“别名”，这些id在 IoC 容器中必须唯一。Bean id的命名方式和命名约定如下：

Bean id 的命名方式：

- **配置全限定类名，唯一**
- **指定id，唯一**
- **指定name，唯一**
- **指定id和name，唯一**
- **指定多个name，唯一**
- **指定别名，唯一**

Bean id 的命名约定：

1. **遵循XML命名规范**
2. **由字母，数字，下划线组成**
3. **驼峰式，首个单词字母大写**

Spring Bean 的实例化

Spring Bean 的实例化

Spring IoC容器如何实例化Bean呢？传统应用程序可以通过new和反射方式进行实例化Bean。而Spring IoC 容器则需要根据Bean定义里的配置元数据使用反射机制来创建Bean。在Spring IoC 容器中主要有以下几种创建Bean实例的方式：

- 使用构造器实例化Bean
- 使用静态工厂方式实例化Bean
- 使用实例工厂方法实例化Bean

Spring Bean 的实例化 – 构造器实例化

构造器实例化 Bean 是最简单的方式，Spring IoC容器既能使用默认空构造器也能使用有参数构造器两种方式创建Bean，如以下方式指定要创建的Bean类型：

- 1. 空构造器实例化：
- **<bean id="helloServiceNoWithArgs" class="com.jike.***.HelloWorldImpl" />**

- 2. 有参数构造器实例化：
- **<bean id="helloServiceWithArgs" class=" com.jike.***.HelloWorldmpl">**
- **<!-- 指定构造器参数 -->**
- **<constructor-arg index="0" value="Hello Spring!"/>**
- **</bean>**

Spring Bean 的实例化 – 静态工厂实例化

使用静态工厂的方式除了指定必须的class属性，还要指定factory-method属性来指定实例化Bean的方法，而且使用静态工厂方法也允许指定方法参数，Spring IoC容器将调用此属性指定的方法来获取Bean，配置如下：

- **<!--使用有参数构造参数-->**
- **<bean id="helloServiceStaticFactory" class="com.jike.***.HelloWorldStaticFactory"**
  factory-method="newInstance">
- **<!-- 指定构造器参数 -->**
- **<constructor-arg index="0" value="Hello Static Factory!"/>**
- **</bean>**

Spring Bean 的实例化 – 实例工厂实例化

使用实例工厂方式不能指定class属性，此时必须使用factory-bean属性来指定工厂Bean，factory-method属性指定实例化Bean的方法，而且使用实例工厂方法允许指定方法参数，方式和使用构造器方式一样，配置如下：

```
<!-- 1、定义实例工厂Bean -->
<bean id="beanInstanceFactory" class="com.jike.***.HelloWorldInstanceFactory" />
<!-- 2、使用实例工厂Bean创建Bean -->
<bean id=“helloWorldInstance" factory-bean="beanInstanceFactory"
                    factory-method="newInstance">
  <constructor-arg index="0" value="Hello  Instance Factory!"></constructor-arg>
</bean>
```

Spring Bean 的作用域

Spring Bean 的作用域

Spring Bean 中所说的作用域，在配置文件中即是“scope”。在面向对象程序设计中一般指对象或变量之间的可见范围。而在Spring容器中是指其创建的Bean对象相对于其他Bean对象的请求可见范围：

- Bean 的作用域类型与配置
- Bean 的作用域示例讲解
- Bean 的自定义作用域

Spring Bean 的作用域 – 作用域的类型与配置

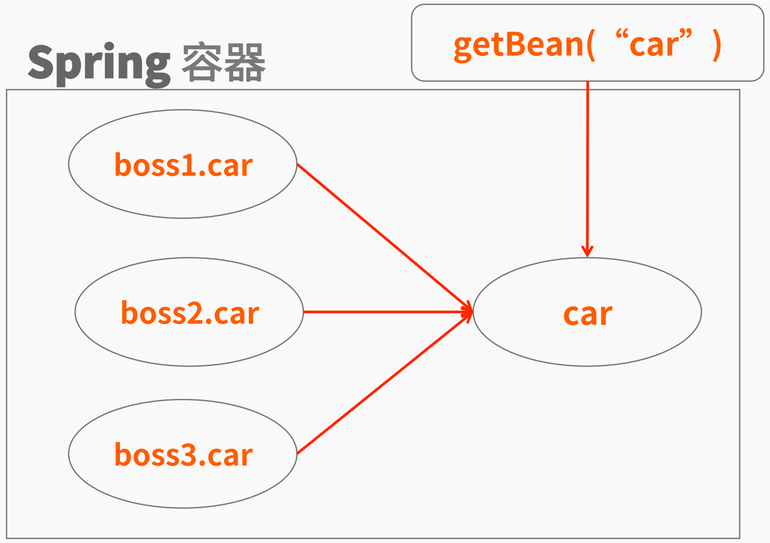
在Spring 容器当中，一共提供了5种作用域类型，在配置文件中，通过属性scope来设置bean的作用域范围。针对不同的作用域，依次示例讲解如下：

- **singleton**：**<bean id="userInfo" class="com.jike.UserInfo" scope="singleton"></bean>**
- **prototype**：**<bean id="userInfo" class="com.jike.UserInfo" scope=" prototype "></bean>**
- **request**：**<bean id="userInfo" class="com.jike.UserInfo" scope=" request "></bean>**
- **session**：**<bean id="userInfo" class="com.jike.UserInfo" scope=" session "></bean>**
- **global session**：**<bean id="userInfo" class="com.jike.UserInfo" scope=“globalSession"></bean>**

Spring Bean 的作用域 – 作用域的类型与配置

singleton作用域是指在Spring IoC容器中仅存在一个Bean的示例，Bean以单实例的方式存在，单实例模式是重要的设计模式之一，在Spring中对此实现了超越，可以对那些非线程安全的对象采用单实例模式：

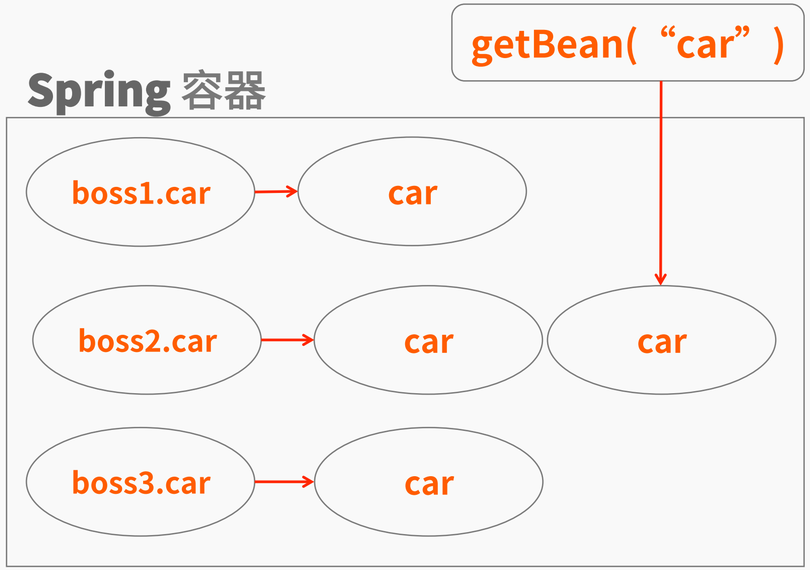
- singleton的配置方式：
- **1. <bean id=“car" class="com.jike.Car" scope="singleton"></bean>**
- **2. <bean id=“boss1" class="com.jike.Boss” p:car-ref=“car"></bean>**
- **3. <bean id=“boss2" class="com.jike.Boss” p:car-ref=“car"></bean>**
- **4. <bean id=“boss3" class="com.jike.Boss” p:car-ref=“car"></bean>**



Spring Bean 的作用域 – 作用域的类型与配置

prototype作用域是指每次从容器中调用Bean时，都返回一个新的实例，即每次调用getBean()时，相当于执行new Bean()的操作。在默认情况下，Spring容器在启动时不实例化prototype的Bean：

- prototype的配置方式：
- 1. <bean id=“car" class="com.jike.Car" scope=“prototype"></bean>
- 2. <bean id=“boss1" class="com.jike.Boss” p:car-ref=“car"></bean>
- 3. <bean id=“boss2" class="com.jike.Boss” p:car-ref=“car"></bean>
- 4. <bean id=“boss3" class="com.jike.Boss” p:car-ref=“car"></bean>



Spring Bean 的作用域 – 作用域的类型与配置

当用户使用Spring的WebApplicationContext时，还可以使用另外3种Bean的作用域，即request,session和globalSession。在使用Web应用环境相关的Bean作用域时，必须在Web容器中进行一些额外的配置：

- 低版本Web容器配置：
- <filter>
- <filter-name>requestContextFilter</filter-name>
- <filter-class>org.springframework.web.filter.RequestContextFilter</filter-class>
- </filter>
- <filter-mapping>
- <filter-name>requestContextFilter</filter-name>
- <servlet-name>/*</servlet-name>
- </filter-mapping>
- 高版本Web容器配置：
- <listener> <listener-class>
- org.springframework.web.context.request.RequestContextListener
- </listener-class></listener>

Web应用环境的作用域：

- **request作用域**
- **session作用域**
- **globalSession作用域**

Spring Bean 的作用域 – 作用域的示例

为了加深大家的理解，如下是编写一个小程序，来具体的了解Bean 的作用域的概念：

- 创建两个Java Bean：老板（Boss.java）以及轿车（Car.java）
- 创建配置文件：
- **<beans … ">**
- ** <bean id="car" class="com.jike.***.Car" scope=“singleton"/>**
- ** <bean id="boss1" class="com.jike.***.Boss" p:car-ref="car" />**
- ** <bean id="boss2" class="com.jike.***.Boss" p:car-ref="car" />**
- ** <bean id="boss3" class="com.jike.***.Boss" p:car-ref="car " />**
- **</beans>**
- 创建测试类进行测试

Spring Bean 的作用域 – 自定义作用域

在Spring 2.0中，Spring的Bean作用域机制是可以扩展的，这意味着，你不仅可以使用Spring提供的预定义Bean作用域，还可以定义自己的作用域，甚至重新定义现有的作用域（不提倡这么做，而且你不能覆盖内置的singleton和prototype作用域）

- 实现自定义Scope类:

  org.springframework.beans.factory.config.Scope

- 注册自定义Scope类:

  ConfigurableBeanFactory.registerScope(String scopeName, Scope scope)

- 使用自定义的Scope:

  Scope customScope = new ThreadScope();

  beanFactory.registerScope(“thread”, customeScope);

  <bean id=“*” class=“***” scope=“scopeName” />**

配置文件的整合

配置文件的整合

多个配置文件：

- Spring-Common.xml位于common文件夹下
- Spring-Connection.xml位于connection文件夹下
- Spring-Module.xml位于module文件夹下

传统加载方式：

```
ApplicationContext context = new ClassPathXmlApplicationContext(new String[]
 {"Spring-Common.xml","Spring-Connection.xml","Spring-ModuleA.xml"});
```

整合配置文件：Spring-All-Module.xml

```
<beans .....>
  <import resource="common/Spring-Common.xml"/>
  <import resource="connection/Spring-Connection.xml"/>
  <import resource="module/Spring-Module.xml"/>
</beans>
```

整合后加载方式：

```
ApplicationContext context = new ClassPathXmlApplicationContext(“Spring-All-Module.xml”);
```

Spring 配置文件浅析

在本套课程中我们学习了Spring 配置文件的相关知识，我们通过简单明了的实例逐步讲解了Spring的配置文件的基本结构，以及如何命名和实例化 Bean，并探讨了Bean的作用域相关知识以及多配置文件整合的概念。你应当掌握了以下知识：

- Spring配置文件的编写
- Bean的命名和实例化
- 针对不同场景为Bean设置相应的作用域
- 多配置文件的意义以及整合方法

通过本课程的学习，你可以运用以上所学的知识为Spring Bean编写简单的配置文件，如果你想继续提高，可以继续在极客学院学习Spring的其他相关课程。

极客学院

jikexueyuan.com

中国最大的IT职业在线教育平台