[**Spring读书笔记-----Spring的Bean之Bean的基本概念**](http://blog.csdn.net/chenssy/article/details/8222744)

标签： [spring](http://www.csdn.net/tag/spring)[Spring](http://www.csdn.net/tag/Spring)[读书笔记](http://www.csdn.net/tag/%e8%af%bb%e4%b9%a6%e7%ac%94%e8%ae%b0)

2012-11-25 16:55 19596人阅读 [评论](http://blog.csdn.net/chenssy/article/details/8222744#comments)(4) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/chenssy/article/details/8222744#report)

http://static.blog.csdn.net/images/category_icon.jpg 分类：

【架构设计】-----SSH框架（61） http://static.blog.csdn.net/images/arrow_triangle%20_down.jpg

版权声明：本文为博主原创文章，未经博主允许不得转载。

        从前面我们知道Spring其实就是一个大型的工厂，而Spring容器中的Bean就是该工厂的产品.对于Spring容器能够生产那些产品，则取决于配置文件中配置。

        对于我们而言，我们使用Spring框架所做的就是两件事：开发Bean、配置Bean。对于Spring矿建来说，它要做的就是根据配置文件来创建Bean实例，并调用Bean实例的方法完成“依赖注入”。

        一、Bean的定义

        <beans…/>元素是Spring配置文件的根元素，<bean…/>元素师<beans../>元素的子元素，<beans…/>元素可以包含多个<bean…/>子元素，每个<bean…/>元素可以定义一个Bean实例，每一个Bean对应Spring容器里的一个Java实例定义Bean时通常需要指定两个属性。

         Id：确定该Bean的唯一标识符，容器对Bean管理、访问、以及该Bean的依赖关系，都通过该属性完成。Bean的id属性在Spring容器中是唯一的。

        Class：指定该Bean的具体实现类。注意这里不能使接口。通常情况下，Spring会直接使用new关键字创建该Bean的实例，因此，这里必须提供Bean实现类的类名。

        下面是定义一个Bean的简单配置

**[html]** [view plain](http://blog.csdn.net/chenssy/article/details/8222744) [copy](http://blog.csdn.net/chenssy/article/details/8222744)

[print](http://blog.csdn.net/chenssy/article/details/8222744)[?](http://blog.csdn.net/chenssy/article/details/8222744)

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. **<beans** xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3. xmlns="http://www.springframework.org/schema/beans"
4. xsi:schemaLocation="http://www.springframework.org/schema/beans
5. http://www.springframework.org/schema/beans/spring-beans-3.0.xsd"**>**
6. <!-- 定义第一个Bean实例：bean1 -->
7. **<bean** id="bean1" class="com.Bean1" **/>**
9. <!-- 定义第二个Bean实例：bean2 -->
10. **<bean** id="bean2" class="com.Bean2" **/>**
12. **</bean>**

        Spring容器集中管理Bean的实例化，Bean实例可以通过BeanFactory的getBean(Stringbeanid)方法得到。BeanFactory是一个工厂，程序只需要获取BeanFactory引用，即可获得Spring容器管理全部实例的引用。程序不需要与具体实例的实现过程耦合。大部分Java EE应用里，应用在启动时，会自动创建Spring容器，组件之间直接以依赖注入的方式耦合，甚至无须主动访问Spring容器本身。

        当我们在配置文件中通过<bean id=”xxxx” class=”xx.XxClass”/>方法配置一个Bean时，这样就需要该Bean实现类中必须有一个无参构造器。故Spring底层相当于调用了如下代码：

**[html]** [view plain](http://blog.csdn.net/chenssy/article/details/8222744) [copy](http://blog.csdn.net/chenssy/article/details/8222744)

[print](http://blog.csdn.net/chenssy/article/details/8222744)[?](http://blog.csdn.net/chenssy/article/details/8222744)

1. Xxx = new xx.XxClass()

        如果在配置文件中通过构造注入来创建Bean：

**[html]** [view plain](http://blog.csdn.net/chenssy/article/details/8222744) [copy](http://blog.csdn.net/chenssy/article/details/8222744)

[print](http://blog.csdn.net/chenssy/article/details/8222744)[?](http://blog.csdn.net/chenssy/article/details/8222744)

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. **<beans** xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3. xmlns="http://www.springframework.org/schema/beans"
4. xsi:schemaLocation="http://www.springframework.org/schema/beans
5. http://www.springframework.org/schema/beans/spring-beans-3.0.xsd"**>**
6. **<bean** id="bean1" class="com.Bean1"**>**
7. **<constructor-arg** value="chenssy"**/>**
8. **<constructor-arg** value="35-354"**/>**
9. **</bean>**
11. **</beans>**

        则Spring相当于调用如下代码：

**[html]** [view plain](http://blog.csdn.net/chenssy/article/details/8222744) [copy](http://blog.csdn.net/chenssy/article/details/8222744)

[print](http://blog.csdn.net/chenssy/article/details/8222744)[?](http://blog.csdn.net/chenssy/article/details/8222744)

1. Bean bean = new com.Test("chenssy","35-354");

        除了可以为<bean…/>元素指定一个id属性外，还可以为<bean…/>元素指定name属性，用于为Bean实例指定别名。如果需要为Bean实例指定多个别名，可以在name属性中使用逗号、冒号或者空格来分隔多个别名，后面通过任一别名即可访问该Bean实例。但是在一些特殊的情况下，程序无法在定义Bean时就指定所有的别名，而是在其他地方为一个已经存在的Bean实例指定别名，则可以使用<alias…/>元素来完成，该元素有如下两个属性：

        name：该属性指定一个Bean实例的标识名，表示将会为该Bean指定别名。

        alias：指定一个别名.

        如：

**[html]** [view plain](http://blog.csdn.net/chenssy/article/details/8222744) [copy](http://blog.csdn.net/chenssy/article/details/8222744)

[print](http://blog.csdn.net/chenssy/article/details/8222744)[?](http://blog.csdn.net/chenssy/article/details/8222744)

1. **<alias** name=”bean1” alias=”name1”**/>**
2. **<alias** name=”bean2” alias=”name2”**/>**

        在默认情况下，当Spring创建ApplicationContext容器时，Spring会自动预初始化容器中所有的singleton实例，如果我们想让Spring容器预初始化某个singleton Bean，则可以为该<bean…/>元素增加lazy-init属性，该属性用于指定该Bean实例的预初始化，如果设置为true，则Spring不会预初始化该Bean实例。

**[html]** [view plain](http://blog.csdn.net/chenssy/article/details/8222744) [copy](http://blog.csdn.net/chenssy/article/details/8222744)

[print](http://blog.csdn.net/chenssy/article/details/8222744)[?](http://blog.csdn.net/chenssy/article/details/8222744)

1. **<bean** id=”person” class=”com.Person” lazy-init=”true”**/>**

        一、       容器中Bean的作用域        当通过Spring容器创建一个Bean实例时，不仅可以完成Bean实例的实例化，还可以为Bean指定特定的作用域。

        Spring支持5种作用域：

           Singleton：单例模式。在整个SpringIoC容器中，使用singleton定义的Bean将只有一个实例。

           Prototype：原型模式。每次通过容器的getBean方法获取prototype定义的Bean时，都将产生一个新的Bean实例。

           request：对于每次HTTP请求，使用request定义的Bean都将产生一个新的实例，即每次HTTP请求都会产生不同的Bean实例。当然只有在WEB应用中使用Spring时，该作用域才真正有效。

        session：对于每次HTTPSession，使用session定义的Bean都将产生一个新的实例时，即每次HTTP Session都将产生不同的Bean实例。同HTTP一样，只有在WEB应用才会有效。

        global session：每个全局的HTTPSession对应一个Bean实例。仅在portlet Context的时候才有效。

        比较常用的singleton和prototype。如果一个Bean实例被设置为singleton，那么每次请求该Bean时都会获得相同的实例。容器负责跟踪Bean实例的状态，负责维护Bean实例的生命周期行为。如果一个Bean实例被设置为prototype，那么每次请求该di的Bean，Spring都会创建一个新的Bean实例返回给程序，在这种情况下，Spring容器仅仅使用new关键字创建Bean实例，一旦创建成功，容器将不会再跟踪实例，也不会维护Bean实例的状态。

        如果我们不指定Bean的作用域，则Spring会默认使用singleton作用域。

        Java在创建Java实例时，需要进行内存申请。销毁实例时，需要完成垃圾回收。这些工作都会导致系统开销的增加。因此，prototype作用域Bean的创建、销毁代价会比较大。而singleton作用域的Bean实例一旦创建成功，可以重复使用。因此，除非必要，否则尽量避免将Bean的作用域设置为prototype。

        设置Bean的作用域是通过scope属性来指定。可以接受Singleton、prototype、request、session、global session 5个值。

**[html]** [view plain](http://blog.csdn.net/chenssy/article/details/8222744) [copy](http://blog.csdn.net/chenssy/article/details/8222744)

[print](http://blog.csdn.net/chenssy/article/details/8222744)[?](http://blog.csdn.net/chenssy/article/details/8222744)

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. **<beans** xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3. xmlns="http://www.springframework.org/schema/beans"
4. xsi:schemaLocation="http://www.springframework.org/schema/beans
5. http://www.springframework.org/schema/beans/spring-beans-3.0.xsd"**>**
6. <!-- 配置一个singleton Bean实例：默认 -->
7. **<bean** id="bean1" class="com.Bean1" **/>**
8. <!-- 配置一个prototype Bean实例 -->
9. **<bean** id="bean2" class="com.Bean2" scope="prototype"**/>**
11. **</beans>**

        上面的配置，对于bean1没有指定scope属性，则默认使用singleton，而bean2则指定一个prototype。

        测试代码：

**[java]** [view plain](http://blog.csdn.net/chenssy/article/details/8222744) [copy](http://blog.csdn.net/chenssy/article/details/8222744)

[print](http://blog.csdn.net/chenssy/article/details/8222744)[?](http://blog.csdn.net/chenssy/article/details/8222744)

1. **public** **class** SpringTest {
3. **public** **static** **void** main(String[] args) {
4. ApplicationContext ctx = **new** ClassPathXmlApplicationContext("bean.xml");
5. //判断两次请求singleton作用域的Bean实例是否相等
6. System.out.println(ctx.getBean("bean1")==ctx.getBean("bean1"));
7. //判断两次请求prototype作用域的Bean实例是否相等
8. System.out.println(ctx.getBean("bean2")==ctx.getBean("bean2"));
9. }
11. }

        程序运行结果如下

     true

     false

        从上面的运行结果可以看出：对于singleton作用域的Bean，每次请求该id的Bean时都将返回同一个Bean实例，但是prototype返回的都是一个新的Bean实例，每次请求返回的Bean实例都将不同。

        对于request作用域而言，先看如下Bean实例定义：

**[html]** [view plain](http://blog.csdn.net/chenssy/article/details/8222744) [copy](http://blog.csdn.net/chenssy/article/details/8222744)

[print](http://blog.csdn.net/chenssy/article/details/8222744)[?](http://blog.csdn.net/chenssy/article/details/8222744)

1. **<bean** id=”login” class=”com.app.LoginAction” scope=”request”**/>**

        对于每次HTTP请求，Spring容器都会根据login Bean定义创建一个全新的LoginAction Bean实例，且该loginAction Bean实例仅在当前HTTP Request内有效。

        对于session作用域相同。只不过有效范围不同而已。

        request和session作用域只在web应用中才会有效，并且必须在Web应用中增加额外配置才会生效。为了能够让request和session两个作用域生效，必须将HTTP请求对象绑定到位该请求提供的服务线程上，这使得具有request和session作用的Bean实例能够在后面的调用链中被访问到。

        因此我们可以采用两种配置方式：采用Listener配置或者采用Filter配置，在web.xml中。

        Listener配置：

**[html]** [view plain](http://blog.csdn.net/chenssy/article/details/8222744) [copy](http://blog.csdn.net/chenssy/article/details/8222744)

[print](http://blog.csdn.net/chenssy/article/details/8222744)[?](http://blog.csdn.net/chenssy/article/details/8222744)

1. **<listener>**
2. **<listener-class>**
3. org.springframework.web.context.request.RequestContextListener
4. **</listener-class>**
5. **</listener>**

        Filter配置

**[html]** [view plain](http://blog.csdn.net/chenssy/article/details/8222744) [copy](http://blog.csdn.net/chenssy/article/details/8222744)

[print](http://blog.csdn.net/chenssy/article/details/8222744)[?](http://blog.csdn.net/chenssy/article/details/8222744)

1. **<filter>**
2. **<filter-name>**requestContextFilter**</filter-name>**
3. **<filter-class>**org.springframework.web.filter.RequestContextFilter**</filter-class>**
4. **</filter>**
5. **<filter-mapping>**
6. **<filter-name>**requestContextFilter**</filter-name>**
7. **<url-pattern>**/\***</url-pattern>**
8. **</filter-mapping>**

        一旦在web.xml中增加上面两种配置中的一种，程序就可以在Spring配置文件中使用request或者session作用域了。如下：

**[html]** [view plain](http://blog.csdn.net/chenssy/article/details/8222744) [copy](http://blog.csdn.net/chenssy/article/details/8222744)

[print](http://blog.csdn.net/chenssy/article/details/8222744)[?](http://blog.csdn.net/chenssy/article/details/8222744)

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. **<beans** xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3. xmlns="http://www.springframework.org/schema/beans"
4. xsi:schemaLocation="http://www.springframework.org/schema/beans
5. http://www.springframework.org/schema/beans/spring-beans-3.0.xsd"**>**
6. <!-- 指定使用request作用域 -->
7. **<bean** id="p" class="com.app.Person" scope="request"**/>**
9. **</beans>**

        上面的配置文件配置了一个实现类Person的Bean，指定它的作用域为request。这样Spring容器会为每次的HttP请求生成一个Person的实例，当该请求响应结束时，该实例也会被注销。