# 一、基于XML的配置

  采用Schema格式

<?xml version="1.0" encoding="UTF-8" ?>  
<beans xmlns="http://www.springframework.org/schema/beans"//默认命名空间，用于bean的定义  
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"//xsi标准命名空间，用于为每个文档指定相对应的Schema样式文件  
   xmlns:p="http://www.springframework.org/schema/p"  
   xmlns:util="http://www.springframework.org/schema/util"  
   xmlns:aop="http://www.springframework.org/schema/aop" //自定义的一种命名空间   
   xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.0.xsd http://www.springframework.org/schema/util http://www.springframework.org/schema/util/spring-util-4.0.xsd">  
     
   <bean id="car" class="com.smart.fb.Car"  
         p:brand="红旗CA72" p:maxSpeed="200" p:price="20000.00"/>  
   ......  
   </beans>

Schema在文档根节点中通过xmlns对文档所引用的命名空间进行声明。文档后面的元素可通过命名空间别名加以区分。<aop:config/>

# 二、Bean基本配置

<bean id="car" class="com.smart.fb.Car">

id为这个Bean的名称，通过容器的getBean("car")即可获取对应的Bean，class定义了bean的实现类。

id的命名必须以字母开始，不能以逗号和空格结尾。但是name属性没有限制。

spring不允许有两个id相同的bean，但是可以name相同。返回是返回最后一个。

# 依赖注入

1.属性注入：要求bean提供一个默认的构造函数，并有相应的Setter方法。spring是先调用Bean的默认构造函数实例化Bean对象，然后通过反射调用set方法注入属性值。但是spring只会检查是否有应的set方法，而有没有该属性不关注。

在xml中

<bean id="boss" class="com.smart.attr.Boss">  
   <property name="car" ref="car" />  
   <property name="name" value="Tom" />  
   <property name="age" value="45" />  
</bean>

命名规范：xxx属性对于setXxx()方法。变量前两个字母要么都大写要么都小写。

2.构造函数注入

<!--构造函数注入：type -->  
<bean id="car1" class="com.smart.ditype.Car">  
   <constructor-arg type="java.lang.String">  
      <value>红旗CA72</value>  
   </constructor-arg>  
   <constructor-arg type="double">  
      <value>20000</value>  
   </constructor-arg>  
</bean>  
  
<!--  
   构造函数注入：index <bean id="car2" class="com.smart.ditype.Car">  
   <constructor-arg index="0" value="红旗CA72" /> <constructor-arg  
   index="1" value="中国一汽" /> <constructor-arg index="2" value="20000" />  
   </bean>  
-->  
  
<!--构造函数注入：type&index -->  
<bean id="car3" class="com.smart.ditype.Car">  
   <constructor-arg index="0" type="java.lang.String">  
      <value>红旗CA72</value>  
   </constructor-arg>  
   <constructor-arg index="1" type="java.lang.String">  
      <value>中国一汽</value>  
   </constructor-arg>  
   <constructor-arg index="2" type="int">  
      <value>200</value>  
   </constructor-arg>  
</bean>  
<bean id="car4" class="com.smart.ditype.Car">  
   <constructor-arg index="0">  
      <value>红旗CA72</value>  
   </constructor-arg>  
   <constructor-arg index="1">  
      <value>中国一汽</value>  
   </constructor-arg>  
   <constructor-arg index="2" type="int">  
      <value>200</value>  
   </constructor-arg>  
</bean>  
  
<!--构造函数注入：自动识别入参类型 -->  
<bean id="boss1" class="com.smart.ditype.Boss">  
   <constructor-arg>  
      <value>John</value>  
   </constructor-arg>  
   <constructor-arg>  
      <ref bean="car" />  
   </constructor-arg>  
   <constructor-arg>  
      <ref bean="office" />  
   </constructor-arg>  
</bean>  
<bean id="office" class="com.smart.ditype.Office" />

# 四、注入参数

1. 字面值：基本数据类型及其封装类、String类都可以采用字面值注入。通过value=""或者<value>方式来注入。<![CDATA[]]>作用是让XML解析器把[]内的字符串当成普通文本对待。一般情况下XML会忽略标签内部字符串的前后空格，但是在spring中不会。
2. 引用其他bean：通过ref元素。<property name="car" ref="car"></property>或者<ref bean="car"/> bean是同一容器或者父容器中的bean，local只能引用同一配置文件中的bean，parent引用父容器的bean
3. 内部bean：类似内部类，没有名字不能被外部引用。
4. null值：<value></value>这样会被解析成空字符串。<value><null/></value>这样才是空。
5. 集合类型属性

<list>  
   <value>看报</value>  
   <value>赛车</value>  
   <value>高尔夫</value>  
   </list>  
     
<set>  
   <value>看报</value>  
   <value>赛车</value>  
   <value>高尔夫</value>  
</set>  
  
<map>  
   <entry >  
      <key>  
         <value>AM</value>  
      </key>  
      <value>会见客户</value>  
   </entry>  
   <entry>  
      <key>  
         <value>PM</value>  
      </key>  
      <value>公司内部会议</value>  
   </entry>  
</map>  
  
<props>  
   <prop key="jobMail">john-office@smart.com</prop>  
   <prop key="lifeMail">john-life@smart.com</prop>  
</props>  
  
<bean id="parentBoss"  abstract="true"  
   class="com.smart.attr.Boss">  
   <property name="favorites">  
      <set>  
         <value>看报</value>  
         <value>赛车</value>  
         <value>高尔夫</value>  
      </set>  
   </property>  
</bean>  
<bean id="childBoss"  parent="parentBoss">  
   <property name="favorites">  
      <set merge="true">  
         <value>爬山</value>  
         <value>游泳</value>  
      </set>  
   </property>  
</bean>

子bean的favorites最后有5个元素。merge="true"是和父bean的同名进行集合属性合并。

# 五、bean之间的关系

1.继承：子bean继承父bean。重复的会覆盖。

1）父bean：abstract="true"

2）子bean：parent="parbean"

   <!-- 父子<bean> -->  
<bean id="abstractCar" class="com.smart.tagdepend.Car"  
      p:brand="红旗CA72" p:price="2000.00" p:color="黑色"  
      abstract="true"/>  
        
<bean id="car3" parent="abstractCar">  
   <property name="color" value="红色"/>  
</bean>  
<bean id="car4" parent="abstractCar" >  
   <property name="color" value="白色"/>  
</bean>

2.引用

# 六、整合多个配置文件

通过<import>将多个配置文件引到一个文件中，进行配置文件集成。

<import resource="classpath:com/.../bean.xml"/>

# 七、Bean的作用域

1. siglenton作用域：默认情况下，spring的ApplicationContext容器在启动时，会自动实例化所有的sigleton的bean并缓存在容器中。如果不希望提前实例化，可以通过lazy-init="true"来等到使用是才实例化。
2. prototype作用域：scope="prototype"指定非单例作用域的Bean。
3. web环境下：在低版本的web，可以使用http请求过滤器进行配置，高版本可以使用http请求监听器来进行配置。

1）reqest作用域：每次http请求就会调用。

2）session：横跨整个Session，Session中所有http请求共享一个bean。

3）glabalSession

1. <bean ..><aop:scoped-proxy/></bean>则注入的是动态代理对象。

# 八、基于注解的配置

1. 使用注解定义Bean：@Component表示该类为Bean。@Repository：对DAO实现类的标注。
2. 扫描注解定义的Bean：

第一步：声明context命名空间

xmlns:context="http://www.springframework.org/schema/context"

  第二步：使用component-scan的base-package属性指定一个需扫描的包。

<context:component-scan base-package="com.smart.anno">  
    <context:include-filter type="aspectj" expression="com.smart.anno.\*Plugin+"/>  
    <context:include-filter type="aspectj" expression="com.smart.anno.MyComponent"/>  
    <context:exclude-filter type="aspectj" expression="com.smart..\*Controller+"/>  
</context:component-scan>

user-default-filters属性默认是扫描@Repository、@Service 和 @Controlle、@Component 除非设为false。

值得注意的是 <context:component-scan/> 配置项不但启用了对类包进行扫描以实施注释驱动 Bean 定义的功能，同时还启用了注释驱动自动注入的功能（即还隐式地在内部注册了 AutowiredAnnotationBeanPostProcessor 和 CommonAnnotationBeanPostProcessor），因此当使用 <context:component-scan/> 后，就可以将 <context:annotation-config/> 移除了。

3.自动装配Bean

1. @Autowired默认按类型匹配方式来找bean，找不到的时候会报异常，加上@Autowired(reqired=false)就不会抛出异常了。加上@Qualifier注解限定Bean的名称。
2. 对类方法进行标注：Spring允许对方法入参标注@Qualifier以指定注入Bean的名称。
3. 延迟注入：@Lazy在属性及目标Bean类上同时标注。
4. @Resource注解要求提供一个Bean的名称。二者都可以写在字段
5. <context:component-scan
6. base-**package**="com.raylee.my\_new\_spring.my\_new\_spring.ch02.topic\_1\_20">
7. <context:include-filter type="assignable"
8. expression="com.raylee.my\_new\_spring.my\_new\_spring.ch02.topic\_1\_20.Person" />
9. </context:component-scan>

（1）我们配置自动扫描<context:component-scan/>

（2）在自动扫描里面我们在配置<context:include-filter/>，然后type=“assignable”，这里的意思是，Person接口所派生出来的所有类，都自动注册bean

type的四种参数：

assignable-指定扫描某个接口派生出来的类

annotation-指定扫描使用某个注解的类

aspectj-指定扫描AspectJ表达式相匹配的类

custom-指定扫描自定义的实现了org.springframework.core.type.filter.TypeFilter接口的类

regex-指定扫描符合正则表达式的类