# Spring入门

## Bean上下文

这个和hibernate的SessionFactory的获取很类似，要获取其他对象，首先要获取这个根对象，才能衍生出各种对象。

认识2个接口

1. ApplicationContext
2. BeanFactory

ApplicationContext是BeanFactory的子接口，一般都是用他来创建上下文

ApplicationContext的实现类有三个

1. ClassPathXmlApplicationContext:从类加载路径下加载xml
2. FileSystemXmlApplicationContext:从文件系统中指定文件路径加载xml
3. XmlWebAppliactionContext:从web应用中加载xml

|  |
| --- |
| ApplicationContext context=**new** ClassPathXmlApplicationContext("applicationContext.xml");  Person person=(Person)context.getBean("chinese");  person.Speak();  person=(Person)context.getBean("american");  person.Speak(); |

Spring的入门程序绝对是最简单的！

|  |
| --- |
| <bean id=*"chinese"* class=*"com.bean.ChineseImpl"*>  <property name=*"name"*>  <value>小明</value>  </property>  <property name=*"age"*>  <value>10</value>  </property>  </bean>  <bean id=*"american"* class=*"com.bean.AmericanImpl"*>  <property name=*"name"*>  <value>Tom</value>  </property>  <property name=*"age"*>  <value>15</value>  </property>  </bean> |

# 在web中使用spring

## 说明

把包复制到web-inf/lib目录下

然后把bean配置到web.xml的上下文中，就如同在bean中配置数据源一样

|  |
| --- |
| <context-param>  <param-name>contextConfigLocation</param-name>  <param-value>/WEB-INF/classes/applicationContext.xml </param-value>  </context-param> |

如果有多个文件，可以在param\_vaule中用逗号分隔多个文件

对于spring ,hibernate,struts的整合，spring应该是作为最底层的框架的，配置也是最先配置spring

# Bean的装配

## 普通属性

|  |
| --- |
| <bean id=*"girl"* class=*"com.bean.ChineseImpl"*>  <property name=*"name"*>  <value>wife</value>  </property>  <property name=*"age"*>  <value>10</value>  </property>  </bean> |

## Bean装配其他bean

|  |
| --- |
| bean id=*"mycompany"* class=*"com.bean.Company"*>  <property name=*"boss"*>  <ref bean=*"huangning"*/>  </property>  </bean> |

另外需要注意的是，如果是属性注入，就必须给属性添加get/set方法，否则会报错！

## 注解注入

Bean还是需要配置，只不过不再需要填写属性信息了

|  |
| --- |
| <context:component-scan base-package=*"com.spring"* />  <bean id=*"tiger"* class=*"com.animal.Tiger"* />  <bean id=*"mon"* class=*"com.animal.Monkey"* />  <bean id=*"zoo"* class=*"com.animal.Zoo"* /> |

用Autowired可以不用再写get/set方法了！！

如果加上required=false表示，即使属性bean不存在，也不会报错，就显示为null

|  |
| --- |
| **package** com.animal;  **import** org.springframework.beans.factory.annotation.Autowired;  **public** **class** Zoo {  @Autowired(required=**false**)  **private** Tiger tiger;  @Autowired(required=**false**)  **private** Monkey monkey;    **public** String toString(){  **return** tiger + "\n" + monkey;  }    } |

他这个是首先按照类型来的，2个属性类型分别是Tiger和Monkey,只要bean.xml中有这2个类型的bean，就会自动装配，如果有2个类型相同的，那就是按照名字来装配了。

如果容器中有一个以上匹配的Bean，则可以通过@Qualifier注解限定Bean的名称

|  |
| --- |
| @Autowired(required=**false**)  @Qualifier("mon")  **private** Monkey monkey; |

另外一个注解是@Resource,他也可以完成自动装配。

说一下@Resource的装配顺序：  
(1)、@Resource后面没有任何内容，默认通过name属性去匹配bean，找不到再按type去匹配  
(2)、指定了name或者type则根据指定的类型去匹配bean  
(3)、指定了name和type则根据指定的name和type去匹配bean，任何一个不匹配都将报错

然后，区分一下@Autowired和@Resource两个注解的区别：  
(1)、@Autowired默认按照byType方式进行bean匹配，@Resource默认按照byName方式进行bean匹配  
(2)、@Autowired是Spring的注解，@Resource是J2EE的注解，这个看一下导入注解的时候这两个注解的包名就一清二楚了  
Spring属于第三方的，J2EE是Java自己的东西，因此，建议使用@Resource注解，以减少代码和Spring之间的耦合。

结论：尽量使用@Resource注解，这是通用的！！

# 入门例子的最简单化

## 说明

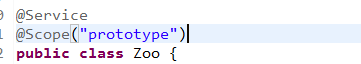
上面这个例子，还可以继续简化，因为spring的配置文件里面还有15行~17行三个bean，下一步的简化是把这三个bean也给去掉，使得spring配置文件里面只有一个自动扫描的标签，增强Java代码的内聚性并进一步减少配置文件。

要继续简化，可以使用@Service。先看一下配置文件，当然是全部删除了：

只留下

|  |
| --- |
| <context:component-scan base-package=*"com.spring"* /> |

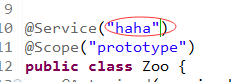
加上@Service



Prototype表示每次都会new一个新的出来，和单利模式singleton是相反的。

默认的bean名字就是类名zoo，首字母小写

也可以指定bean的名字是什么



这样就生成了haha的bean名字了







意思就是只要是定义dao组件，类前面就加上@Repository

注入dao组件的时候还是用Service啦

## 常用注解汇总

**Spring常用注解汇总**  
本文汇总了Spring的常用注解，以方便大家查询和使用，具体如下：

使用注解之前要开启自动扫描功能，其中base-package为需要扫描的包(含子包)。

<context:component-scan base-package="cn.test"/>

@Configuration把一个类作为一个IoC容器，它的某个方法头上如果注册了@Bean，就会作为这个Spring容器中的Bean。  
@Scope注解 作用域  
@Lazy(true) 表示延迟初始化  
@Service用于标注业务层组件、   
@Controller用于标注控制层组件（如struts中的action）  
@Repository用于标注数据访问组件，即DAO组件。  
@Component泛指组件，当组件不好归类的时候，我们可以使用这个注解进行标注。  
@Scope用于指定scope作用域的（用在类上）  
@PostConstruct用于指定初始化方法（用在方法上）  
@PreDestory用于指定销毁方法（用在方法上）  
@DependsOn：定义Bean初始化及销毁时的顺序  
@Primary：自动装配时当出现多个Bean候选者时，被注解为@Primary的Bean将作为首选者，否则将抛出异常  
@Autowired 默认按类型装配，如果我们想使用按名称装配，可以结合@Qualifier注解一起使用。如下：  
@Autowired @Qualifier("personDaoBean") 存在多个实例配合使用  
@Resource默认按名称装配，当找不到与名称匹配的bean才会按类型装配。  
@PostConstruct 初始化注解  
@PreDestroy 摧毁注解 默认 单例  启动就加载  
@Async异步方法调用