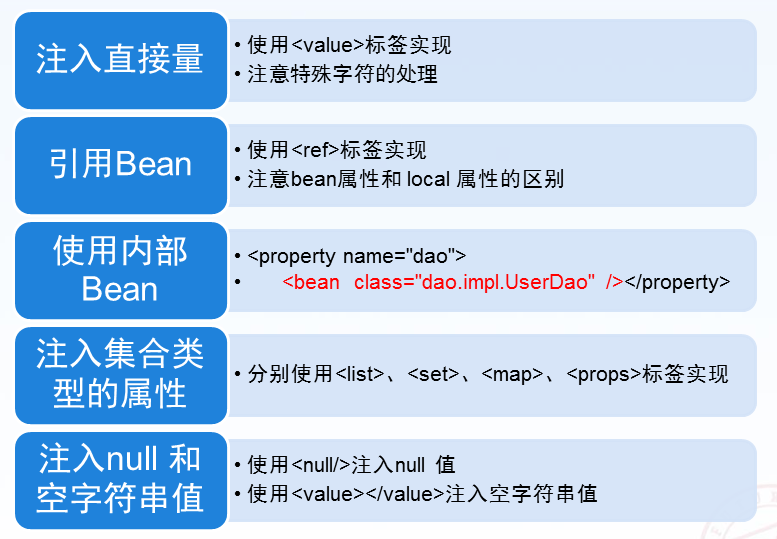
课程目标

1. (掌握)IOC注入不同的数据类型
2. 理解AOP原理(重点 难点)
3. (会)AOP实现
4. 使用依赖注入不同的数据类型



1. 使用value标签注入直接量

|  |
| --- |
| <**property name="xh"**>  <**value**>102</**value**> </**property**> |

1. 使用ref标签引用bean对象

|  |
| --- |
| <**property name="address"**>  *<ref bean="address"></ref>*  ***<!—定义本地bean对象🡪***  **<bean class="com.Address" p:bh="101" p:info="不详"></bean>** </**property**> |

1. 注入集合类型

集合分类:List集合、set集合、map集合、propertes集合(键值集合:键值都字符串)

|  |
| --- |
| *<!--注入List集合-->* <**property name="tels"**>  <**list**>  <**value**>13260601227</**value**>  <**value**>13260601228</**value**>  <**value**>13260601229</**value**>  </**list**> </**property**> *<!--注入Set集合-->* <**property name="adds"**>  <**set**>  <**ref bean="address"**></**ref**>  <**bean class="com.Address" p:bh="104" p:info="不详1"**></**bean**>  <**bean class="com.Address" p:bh="105" p:info="不详2"**></**bean**>  </**set**> </**property**> *<!--注入键值对象集合map-->* <**property name="cjs"**>  <**map**>  *<!--entry表示一个键值对-->* <**entry key="语言" value="90"**></**entry**>  <**entry key="数学" value="80"**></**entry**>  <**entry key="英语" value="10"**></**entry**>  </**map**> </**property**>  *<!--注入键值对象集合properties*  **private** Properties **propcjs**; *//键值对集合*  *-->* <**property name="propcjs"**>  <**props**>  <**prop key="语言"**>90</**prop**>  <**prop key="数学"**>100</**prop**>  <**prop key="英语"**>99</**prop**>  </**props**> </**property**> |

1. 注入空值和空字符串

使用<null/> 表示空对象

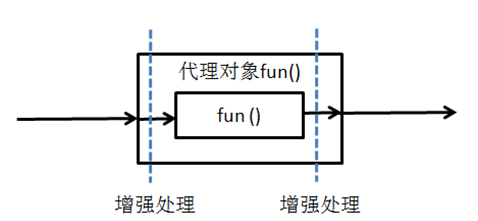
使用<value></value> 表示空字符串

1. Spring的AOP思想 (面向切面编程)

面向切面编程:其实就是对方法进行拦截、实现增强处理

1. **AOP的目标：让我们可以“专心做事”**

1. **AOP原理**
   1. **将复杂的需求分解出不同方面，将散布在系统中的公共功能集中解决**
   2. **采用代理机制组装起来运行，在不改变原程序的基础上对代码段进行增强处理，增加新的功能**



1. 实现AOP面向切面编程的步骤:
   1. 定义增强类
2. 创建前置增强(Advice)

创建一个类实现org.springframework.aop.MethodBeforeAdvice

接口并重写before方法

|  |
| --- |
| *//前置增强类 //代理对象，增强类* **public class** MyBeforeAdvice **implements** MethodBeforeAdvice {  @Override *//重写方法* **public void** before(Method method, Object[] args, Object target) **throws** Throwable {  System.***out***.println(**"method:"**+method.getName());  **for** (Object o:args) {  System.***out***.println(**"参数:"**+o);  }  System.***out***.println(**"target:"**+target.toString());  System.***out***.println(**"我准备开始计算啦...."**);  } } |

1. 定义增强类的bean对象,有可能被拦截的方法的bean对象(拦截的目标对象)

注意:只有spring创建的对象才能进行面向切面

|  |
| --- |
| *<!--目标对象-->* <**bean id="mathTools" class="com.aop.MathTools"**></**bean**> *<!--定义增强类的对象:前置增强-->* <**bean id="mybeforeAdvice" class="com.aop.MyBeforeAdvice"**></**bean**> |

1. 织入切面

|  |
| --- |
| *<!--织入切面配置-->* <**aop:config**>  *<!--定义切入点 指定拦截方法-->* <**aop:pointcut id="pointcut" expression="execution(public int getSum(int,int))"**></**aop:pointcut**>  *<!--指定增强对象-->* <**aop:advisor pointcut-ref="pointcut" advice-ref="myBeforeAdvice"**></**aop:advisor**> </**aop:config**> |

注意:导入aop标签

|  |
| --- |
| **xmlns:aop="http://www.springframework.org/schema/aop"**  **xsi:schemaLocation="http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-4.3.xsd"** |

注意:实现aop切面编程需要导入以下依赖:

|  |
| --- |
| <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-aspects</**artifactId**>  <**version**>4.2.8.RELEASE</**version**>  </**dependency**> |

1. 定义方法切入点

**public int getSum(int,int) 表示方法名getSum带两个整型参数，返回为int**

**public int getSum(..) 表示方法名getSum带任意参数，返回的值int方法**

**\* getSum(..) 表示方法名getSum带任意参数，任意返回值的方法**

**\* \*(..) 表示所有方法**

**\* com.aop.MathTools.\*(..) 表示MathTools类下的所有方法**

**\* com.aop.\*.\*(..) 表示aop包下所有类的所有方法**

**\* com.aop..\*.\*(..) 表示aop包下所有类级其子包下所类的所有方法，任意参数，任意返回值都切**

应用场景：共性集中解决(增强类)

1. 使用后置增强

a.定义后置增强类:创建类实现org.springframework.aop.AfterReturningAdvice 并重写afterReturning方法

|  |
| --- |
| **public class** MyAfterAdvice **implements** AfterReturningAdvice {  @Override  **public void** afterReturning(Object returnValue,  Method method,  Object[] args,  Object target) **throws** Throwable {  System.***out***.println(**"计算结束..."**);  } } |

1. 在spring配置文件中配置增强bean对象

*<!--定义后置增强类的对象-->*<**bean id="myafterAdvice" class="com.aop.MyAfterAdvice"**></**bean**>

1. 配置切入点

|  |
| --- |
| <**aop:advisor advice-ref="myafterAdvice" pointcut-ref="pointcut"**></**aop:advisor**> |

扩展:

1. java的属性文件(理解为文件型的键值对集合)
   1. 扩展名是 properties
   2. 作用:充当java配置文件出现
   3. 存储格式

键=值

键=值

注意:键和值只能为字符串

示例:jdbc.properties

|  |
| --- |
| **jdbc.url**=**jdbc:mysql://localhost:3306/db jdbc.driverClass**=**com.jdbc.mysql.Driver jdbc.username**=**root jdbc.password**=**123** |

* 1. 使用properties对象读取属性文件中的内容

|  |
| --- |
| **public static void** main(String[] args) **throws** IOException {  *//使用Properties加载属性文件的内容* Properties properties=**new** Properties();  *//1.加载属性文件中的数据* FileReader fileReader=**new** FileReader(**"E:\\IdeaProjects\\K0501\\springch01\\src\\main\\resources\\jdbc.properties"**);  properties.load(fileReader);  *//2.显示数据* properties.forEach((o, o2) -> {  System.***out***.println(**"键:"**+o+**"值:"**+o2);  }); } |