1. IOC概念

控制反转：把对象的创建和对象之间的调用过程，交给spring进行管理。

1. IOC目的

为降低耦合度

1. IOC底层实现原理

XML解析、工厂模式、反射

1. IOC、DI范例
   1. 使用setter方法进行注入

<bean id="student" class="com.java.entity.Student">  
 <property name="id" value="1"></property>  
 <property name="name" value="李四"></property>  
</bean>

* 1. 使用构造器注入

<bean id="teacher" class="com.java.entity.Teacher">  
 <constructor-arg index="0" type="java.lang.Integer" value="1"></constructor-arg>  
 <constructor-arg index="1" type="java.lang.String" value="男"></constructor-arg>  
 <constructor-arg index="2" type="java.lang.String" value="王老师"></constructor-arg>  
</bean>

* 1. 复杂类型注入

<bean id="school" class="com.java.entity.School">  
 <property name="id" value="1"></property>  
 <property name="teacher" ref="teacher"></property>  
 <property name="student" ref="student"></property>  
</bean>

* 1. 使用静态工厂实例化bean

<bean id="studentFactory" class="com.java.factory.StudentFactory" factory-method="createStudent">  
</bean>

静态工厂方法实现

public static Student createStudent(){  
 return new Student();  
}

Test

Student student = (Student) applicationContext.getBean("studentFactory");

* 1. 使用非静态工厂类实例化bean

<bean id="teachFactory" class="com.java.factory.TeachFactory"></bean>  
<bean id="teacher1" class="com.java.entity.Teacher" factory-method="creatTeacher" factory-bean="teachFactory"></bean>

非静态方法实现

public Teacher creatTeacher(){  
 return new Teacher();  
}

Test

Teacher teacher1 = (Teacher) applicationContext.getBean("teacher1");