## Spring基础

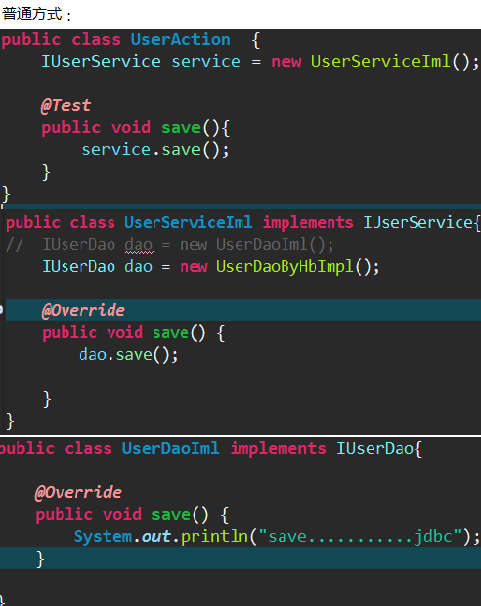
一、Ico(控制反转)

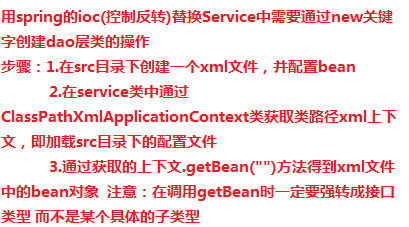
1.导包



文件放在src根目录下，并在src根目录下创建名为applicationContext.xml的xml文件（名字可以任意取）

1. 配置文件







用spring实现类的调用





1. 配置
2. 获取容器的方法

方法一

ApplicationContext c=new ClassPathXmlApplicationContext("applicationContext.xml");

Person p=(Person)bf.getBean("p");

//方法二（此方法已过时）

BeanFactory bf=**new** ~~XmlBeanFactory~~(**new** ClassPathResource("applicationContext.xml"));

Person p=(Person)bf.getBean("p");

区别：

/\*\*

\* 方法一：在容器加载的时候就创建了配置的bean对象

\* 方法二：在容器加载的时候不创建配置的bean对象，在调用getBean()方法时才创建对象

\* 一般推荐第一种方法

\*/

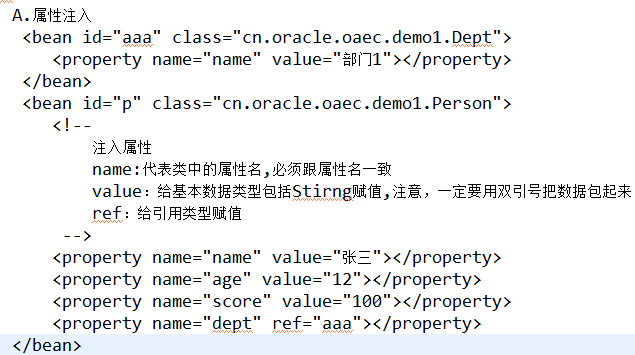
1. Bean详解

spirng 的文件管理，通过import标签

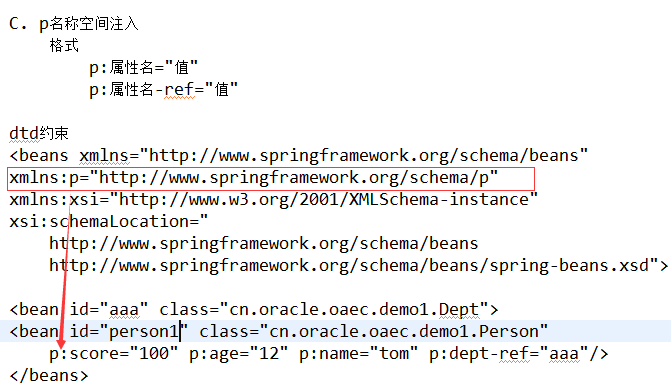
<import resource="cn/oracle/oaec/demo1/entity.xml"/>



## Di注入

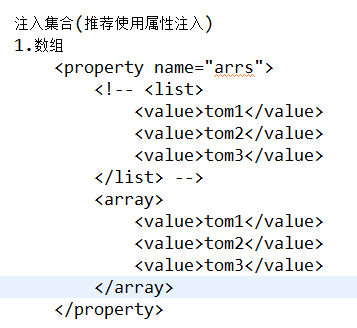


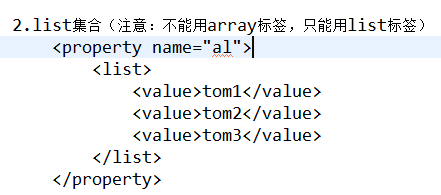


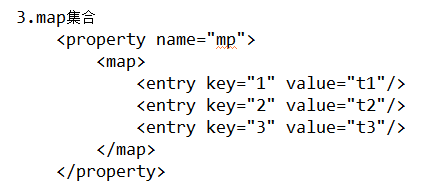


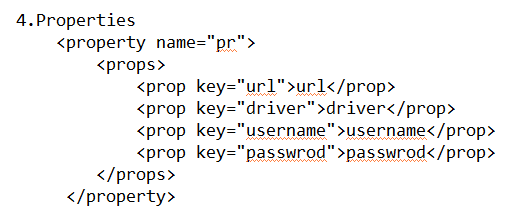


属性注入和构造注入同时用的情况下，结果以属性注入为准







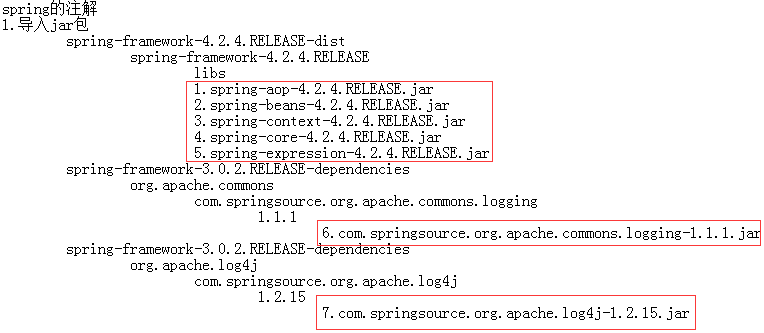


注意点：

spring容器中的每一个bean默认情况下是单例的，就是说只要我们通过同一个id来获取对象，那么都是同一个

但是bean与bean之间不存在单例和多例之分（即使他们的类型一致，属性的值一直，也不存在单例或多例的关系）

1. 注解



1. dtd约束

<!-- dtd约束 -->

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

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

xsi:schemaLocation="

http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans.xsd

http://www.springframework.org/schema/context

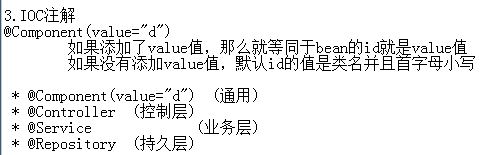
http://www.springframework.org/schema/context/spring-context.xsd">

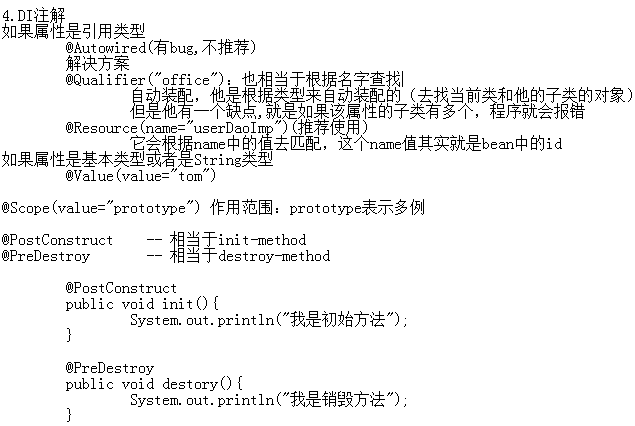
<!-- bean definitions here -->

<!-- 组件扫面 -->

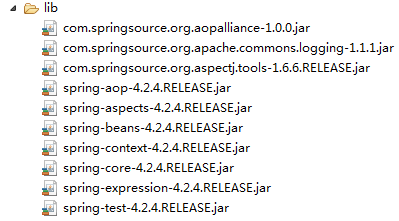
<context:component-scan base-package="cn.oracle.oaec"></context:component-scan>

</beans>





1. AOP面向切面编程
2. 导包



1. 编写applicationContext.xml文件

1）配置方式

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

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

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

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans.xsd

http://www.springframework.org/schema/aop

http://www.springframework.org/schema/aop/spring-aop.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context.xsd">

<!-- 组件扫描 注意：组件扫描一定不要忘记 否则注解就没用了-->

<context:component-scan base-package="zyf"></context:component-scan>

<aop:config>

<aop:aspect ref="trans">//trans为切面的那个类的引用（详见下面注解方式的图）

<aop:before method="beginTransation"

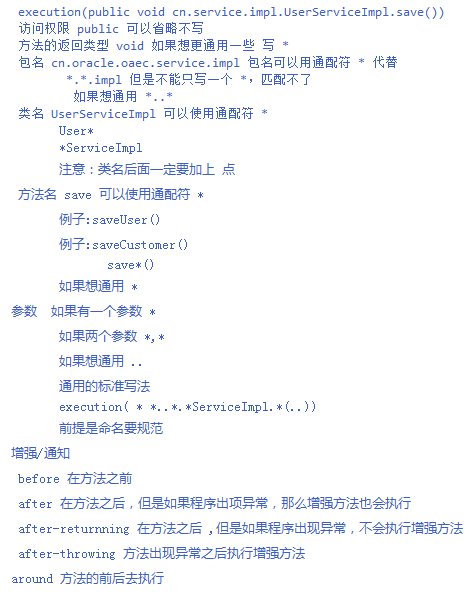
pointcut="execution(\* \*..\*.\*ServiceImpl.\*(..))" />

</aop:aspect>

</aop:config>

</beans>

详细讲解表达式



注意：切面只能切一次，不能重复

如果是**aop:around** 的方法，则方法中必须有一个**ProceedingJoinPoint** 类型的参数，例子：

public void transaction(ProceedingJoinPoint joinPoint){

try {

System.out.println("开启事务...");

/\* Session s=hiernate.getCurrentSession();

\* Transaction tr=s.begingTransaction();

\*/

//就等同于调用cn.oracle.oaec.service.impl.UserServiceImpl.save();

**joinPoint.proceed();**

System.out.println("提交事务...");

/\*

\* tr.commit();

\*/

} catch (Throwable e) {

// TODO Auto-generated catch block

e.printStackTrace();

/\*

\* 日志

\* tr.rollback();

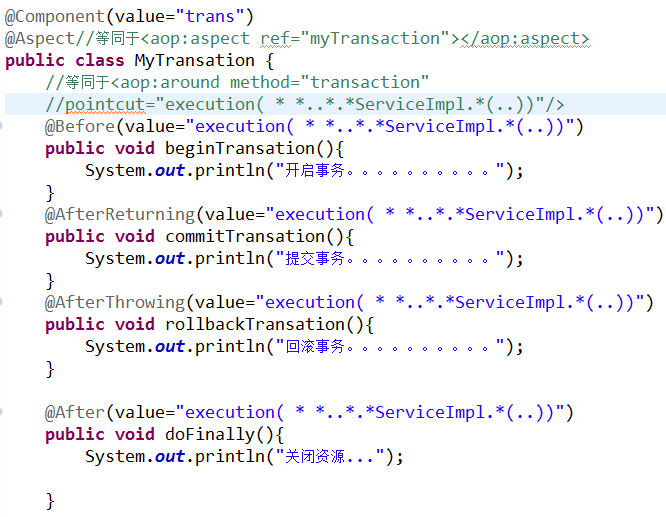
\*/

}

}

1. 注解方式

A.配置切面类



B.在applicationContext.xml配置文件中开启动态代理

<!-- 开启aop的动态代理 -->

<aop:aspectj-autoproxy/>

注意:spring中用到了工厂模式和动态代理模式（暂时只知道这两个）

1. 支持JUnit测试

a.导入支持的jar包： 

b.在类级别上添加注解

@RunWith(SpringJUnit4ClassRunner.**class**)

@ContextConfiguration("classpath:applicationContext.xml")

或者写成：

