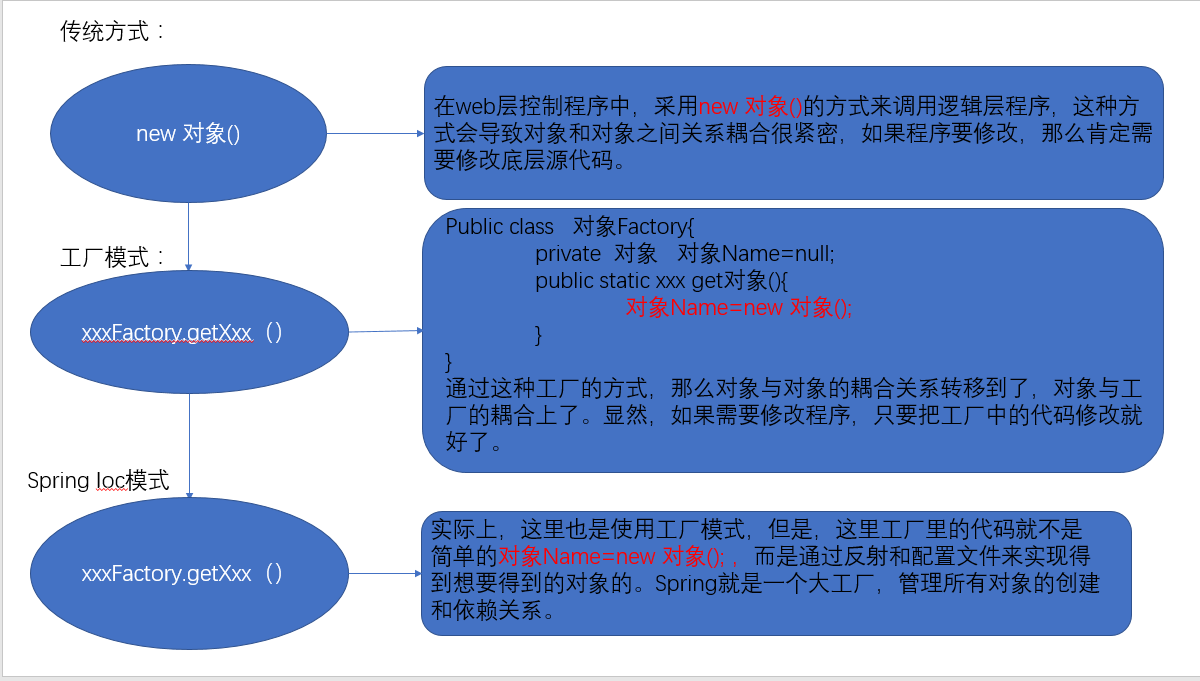
1. Spring的优点

1.1 方便解耦，简化开发：（IOC）

* Spring就是一个大工厂，可以将所有对象创建和依赖关系维护，交给Spring管理。

Web层中的controll中，一般是要调用其他对象，通常任何一个java程序都是要两个及其以上的对象配合完成的，一般来说



1.2. AOP编程的支持

* Spring提供面向切面的编程，可以方便的实现对程序进行：日志记录，性能统计，安全控制，事务处理，异常处理等等
* 主要的意图是：将日志记录，性能统计，安全控制，事务处理，异常处理等代码从业务逻辑代码中划分出来，通过对这些行为的分离，我们希望可以将它们独立到非指导业务逻辑的方法中，进而改  变这些行为的时候不影响业务逻辑的代码。  
  可以通过预编译方式和运行期动态代理实现在不修改源代码的情况下给程序动态统一添加功能的一种技术。

**假设把应用程序想成一个立体结构的话，OOP的利刃是纵向切入系统，把系统划分为很多个模块（如：用户模块，文章模块等等），而AOP的利刃是横向切入系统，提取各个模块可能都要重复操作的部分（如：权限检查，日志记录等等）。由此可见，AOP是OOP的一个有效补充。**

1.3. 其他优点：

1> 声明式事务的支持：

只需要通过配置就可以完成对事务的管理，而无需手动编程。

2> 方便程序的测试

Spring 对 Junit4支持，可以通过注解方便的测试Spring程序。

3> 方便集成各种优秀的框架。

二. Spring中IOC(控制反转)和DI(依赖注入)

2.1 原理

IOC控制反转就是1.1中讲的那样通过配置xml文件，把所有对象的使用，交给spring工厂。这样耦合度很低。

而DI依赖注入，是在IOC基础上的一个概念。

首先，我们要明白java中对象与对象的关系：\* 继承

\* 依赖

\* 聚集

继承好理解，那么依赖是怎么样的一种情况呢？

|  |
| --- |
| public class B{}  public class A{  private B b; //这种情况就叫，A对象依赖B对象  } |

Q：那么spring中的DI 是指什么呢？

A：在我们通过spring工厂建立A对象的时候，spring工厂也会把对像b随着A的创建而创建，这就叫依赖注入。

2.2 IOC 和 DI 的使用

Ioc：

*<?*xml version="1.0" encoding="UTF-8"*?>*<**beans** xmlns="http://www.springframework.org/schema/beans"  
 xmlns:p="http://www.springframework.org/schema/p"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="  
http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd">  
 <**bean** id="demo1" class="springDemo.SpringDemo" >  
 </**bean**>  
</**beans**>

然后在需要使用SpringDemo的时候

//创建一个工厂类  
**ApplicationContext** applicationContext=  
 **new** ClassPathXmlApplicationContext("demoContext.xml");  
**SpringDemo** springDemo= (**SpringDemo**) applicationContext.getBean("demo2");  
springDemo.sayHello();

DI的使用：

spring的依赖注入也是采用配置的方式。

<**bean** id="demo1" class="springDemo.SpringDemo" ></**bean**>  
  
 <!--采用静态工厂来创造-->  
<**bean** id="demo2" class="springDemo.SpringDemoFactory" factory-method="getSpringBean">  
 <**property** name="string" value="周楷"></**property**>  
</**bean**>

<property >这个标签就是用来 填需要注入的属性的，如果需要注入的不是对象，那就可以直接用value来表示值。如果注入的是对象，那么用ref属性。

但，记住，要使用注入的话，那么必须在springDemo中对String string这个属性，添加get，set方法。

三. Spring 细节

3.1 BeanFactory 和 ApplicationContext 区别：

ApplicationContext 继承了 BeanFactory接口

ApplicationContext 一旦加载就会创立xml中所有配置的对象。

BeanFactory 只有在使用 getBean(String id)才会配置需要的对象。

3.2 IOC装配Bean的方式：

装配bean的意思就是创建bean对象的方式，spring分别有三种装配方式：

1> 通过无参数构造

<!--默认是采用无参构造器创造对象的-->  
<**bean** id="demo1" class="springDemo.SpringDemo" ></**bean**>

2> 通过工厂静态构造

<!--采用静态工厂来创造-->  
<**bean** id="demo2" class="springDemo.SpringDemoFactory" factory-method="getSpringBean">  
 <**property** name="string" value="周楷"></**property**>  
</**bean**>

工厂类：

**public class SpringDemoFactory** {  
 **public static SpringDemo** getSpringBean(){  
 // return new SpringDemo("zk");  
 //通过工厂，可以创建有参数的对象。  
 **return new** SpringDemo();  
 }  
}

3> 通过工厂实例创造(了解)

就是通过工厂的非静态方法创建对象。

3.3 bean类的作用范围

spring中bean的作用范围指的是，创造的类是什么样的？是根据bean上的scope元素规定的，**实际开发用的还是singleton、prototype**

<!--创建的是单例模式的bean 这是默认的-->  
<**bean** id="demo3" class="springDemo.SpringDemo" scope="singleton" ></**bean**>  
  
<!--创建的是一般模式的bean,即调用一次就重新创建一个对象-->  
<**bean** id="demo4" class="springDemo.SpringDemo" scope="prototype" ></**bean**>  
  
<!--创建存入request域的bean类-->  
<**bean** id="demo5" class="springDemo.SpringDemo" scope="request" ></**bean**>  
  
<!--创建存入session域的bean类-->  
<**bean** id="demo5" class="springDemo.SpringDemo" scope="session" ></**bean**>  
  
<!--request、session Only valid in the context of a web-aware Spring ApplicationContext.-->

**注意：在spring中，一个类可以有多个bean，只要他们在bean标签配置的id不一样就可以了。**

<**bean** id="demo3" class="springDemo.SpringDemo" scope="singleton" ></**bean**>

<**bean** id="demo4" class="springDemo.SpringDemo" scope="prototype" ></**bean**>

四. Spring的生命周期：