**Spring框架技术**

**安阳师范学院 软件学院**

主讲：郭磊

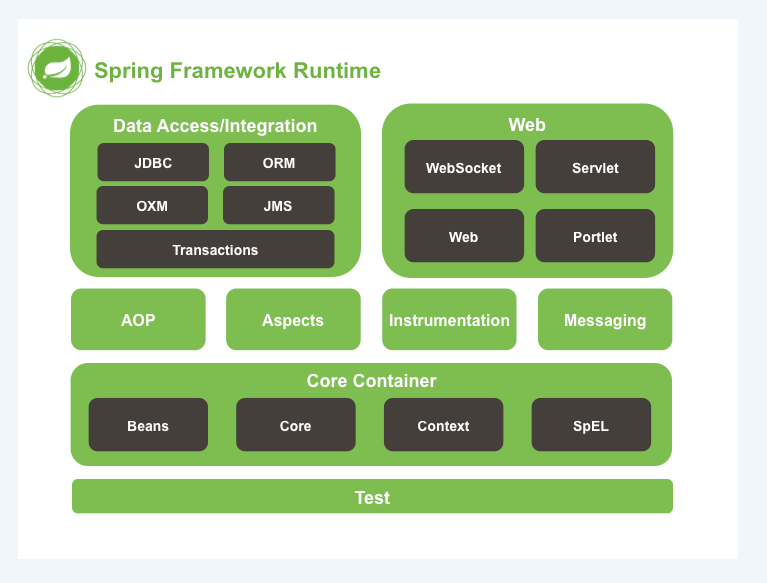
2017年4月

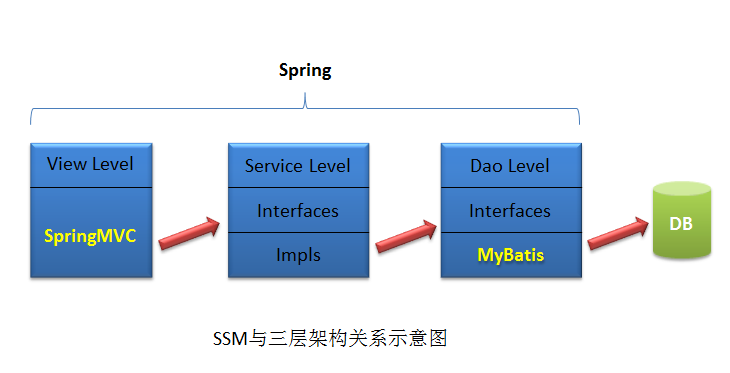
# Spring概述

## Spring简介

Spring是2003年兴起的一个Java轻量级框架。它是了为解决企业级开发的复杂性而创建的一个框架。其是一个分层的Java SE/EE Full-Stack（一站式）框架。

Spring是一个容器，负责管理整个应用中所有的Bean的生命周期，及Bean之间的依赖关系。





## Spring与解耦合

Spring的作用就是为了解耦合，即降低代码间接耦合度。

Spring将应用程序中的代码根据功能划分为了两类：主业务逻辑与系统级服务。

* 主业务逻辑代码的特点：具有很强的专业领域性，复用性相对较低，代码间的耦合度相对较高。
* 系统级服务，也称为交叉业务逻辑，例如，事务管理、日志管理、安全管理等。其特点：专业领域性不强，复用性高，代码间的耦合度相对较低。系统级服务，是为主业务逻辑服务的，其需要交叉（织入）到主业务逻辑代码中。这样的话，主业务逻辑与交叉业务逻辑间的耦合度就比较高了。

Spring针对于以上两种高耦合度代码，使用了IoC与AOP两种技术来降低它们的耦合度。所以，对于Spring的学习重点就是：IoC与AOP。

## Spring特点

（1）非侵入式编程

POJO，Plain Old Java Object

（2）容器

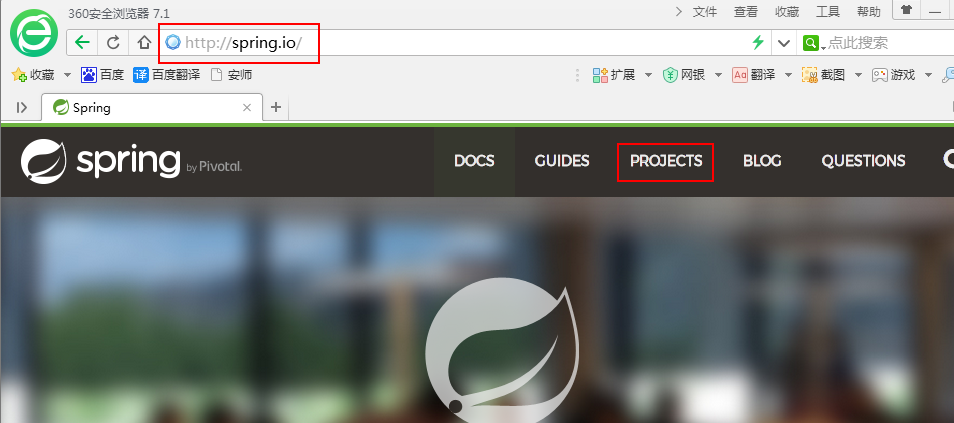
（3）IoC

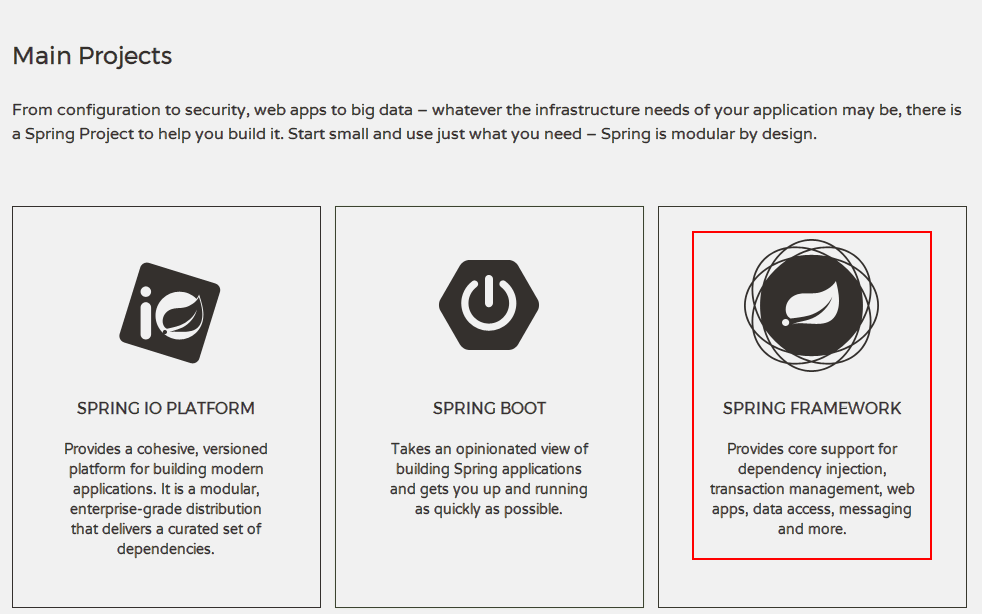
（4）AOP

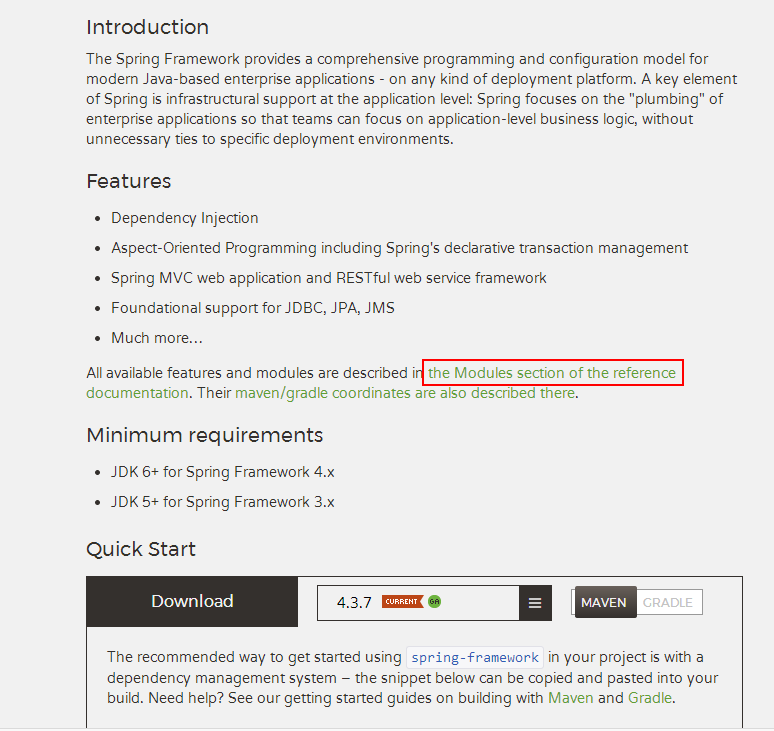
## Spring的下载与目录结构

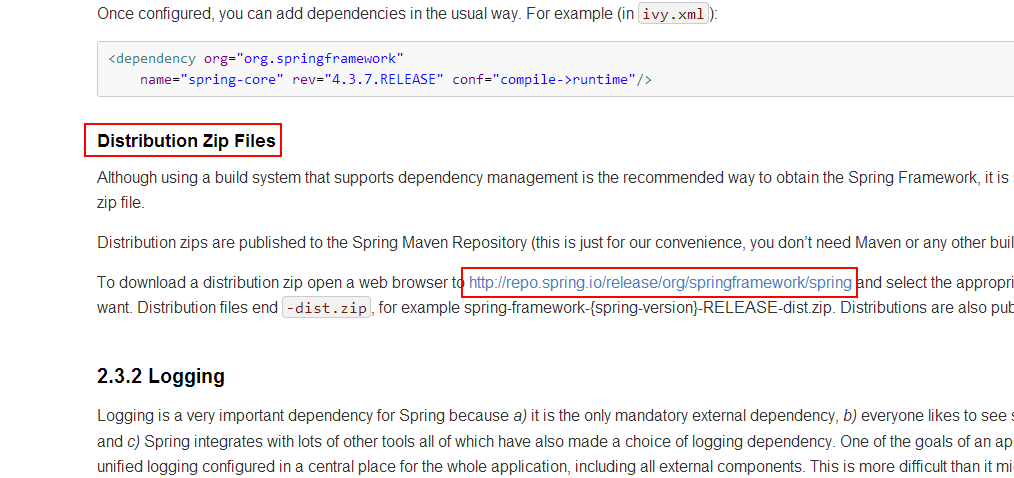
### 下载

http://spring.io

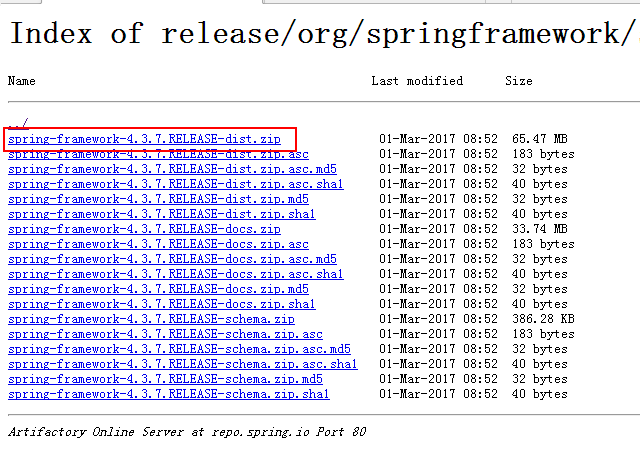




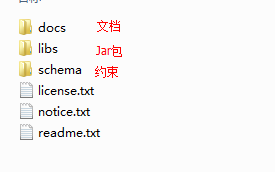








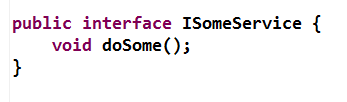
### 结构



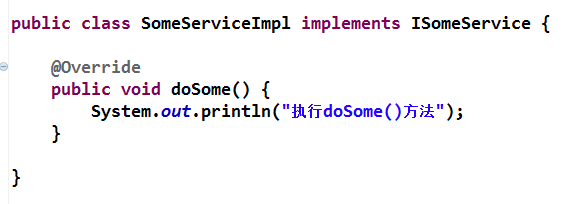
## 第一个Spring程序

### 当前编程方式存在的问题

#### 定义接口



#### 定义接口实现类

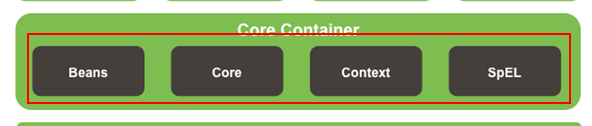


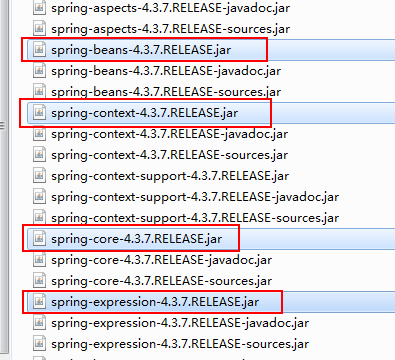
#### 定义测试类



### 使用Spring降低耦合度

#### 导入Jar包





#### 创建Spring容器-配置文件

对于程序员来说，Spring容器就是一个XML配置文件；对于Java代码来说，Spring容器就是一个对象。这个对象，肯定是将这个XML配置文件加载到内存后，形成的对象。

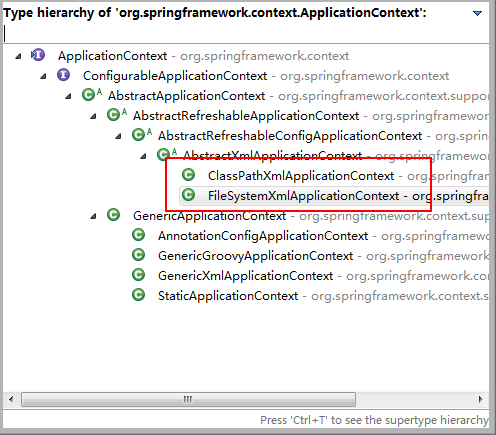
这个XML配置文件约束从Spring框架的如下文件中可以找到：

\docs\spring-framework-reference\html\xsd-configuration.html





#### 修改测试类





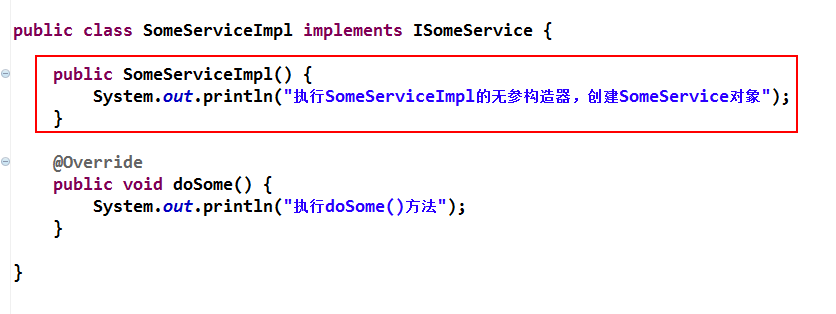
### FileSystemXmlApplicationContext用法

将applicationContext放置到相应位置，修改测试类。

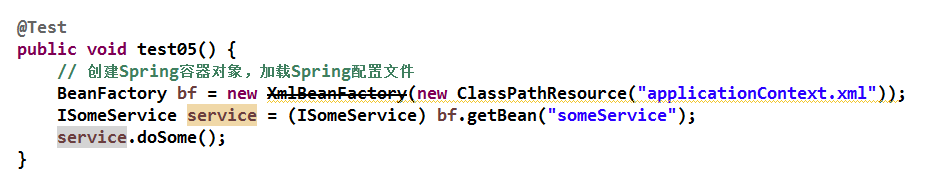


### 两个Spring容器的区别

#### 修改SomeServiceImpl类



#### 修改测试类



#### 区别

ApplicationContext容器：

当容器被创建时，会自动将容器中所有的Bean创建。

优点：系统响应速度快。

不足：浪费系统资源。

BeanFactory容器：

当容器被创建时，并不会自动将容器中的任何Bean创建。只有当真正使用该Bean时，才会由容器创建这个Bean。

优点：节省系统资源。

不足：相对来说，系统响应速度慢。

# Spring与IoC

IoC，Inversion of Control，控制反转。将原本由代码操纵的对象操控权，交由第三方容器，即反转给了第三方容器。这种对象的依赖关系管理方式，称为IoC。

IoC是一种思想，是一个概念。其实现方式很多，较著名的有两种：

DL：Dependency Lookup，依赖查找。其典型应用是JNDI。Java Naming and Directory Interface，Java名称与服务接口。其实JNDI，就是一个第三的容器。

DI：Dependency Injection，依赖注入。是目前最优秀的解耦方式。其典型应用是Spring。

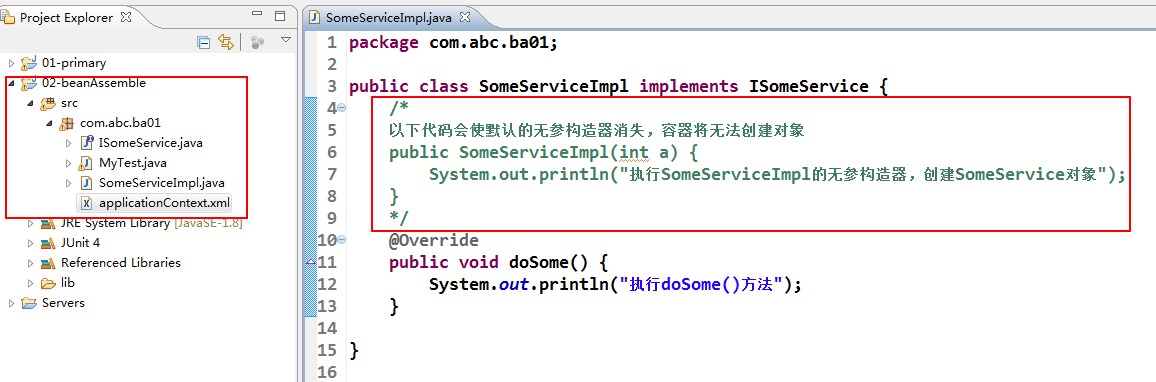
## Bean的装配

Bean的装配，简单来说，就是对象的创建。容器根据代码的要求，创建好对象后，再将对象传递给代码的过程，就是Bean的装配。

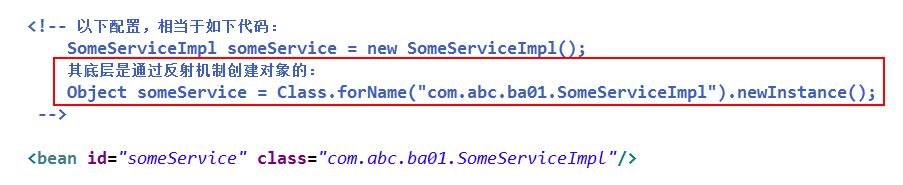
### 默认装配方式

默认装配方式指，容器调用Bean类的无参构造器创建对象。

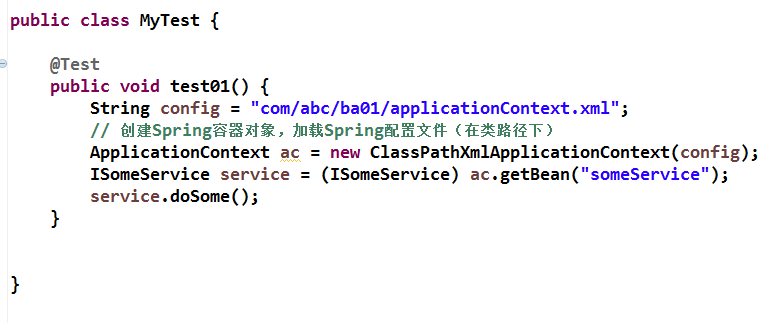
#### 修改接口实现类



#### 修改配置文件

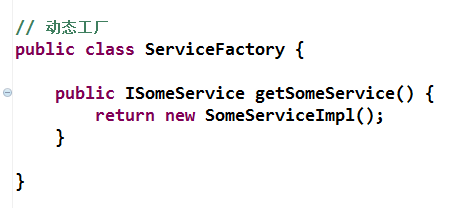


#### 修改测试类

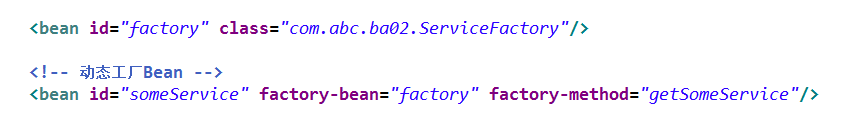


### 动态工厂Bean

#### 定义工厂类



#### 修改配置文件

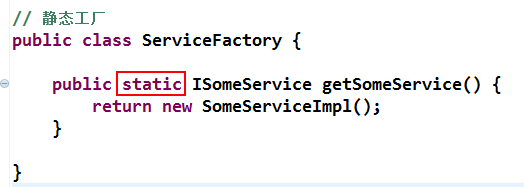


#### 修改测试类

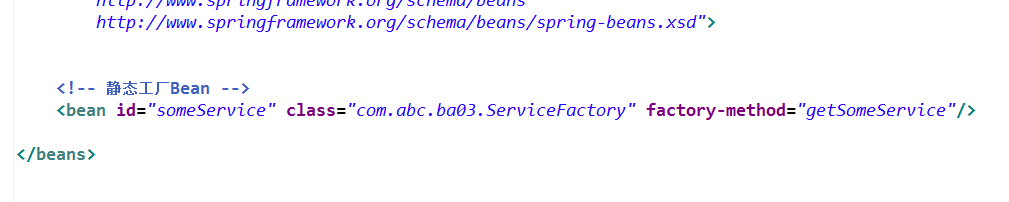


### 静态工厂Bean

#### 修改工厂类



#### 修改配置文件

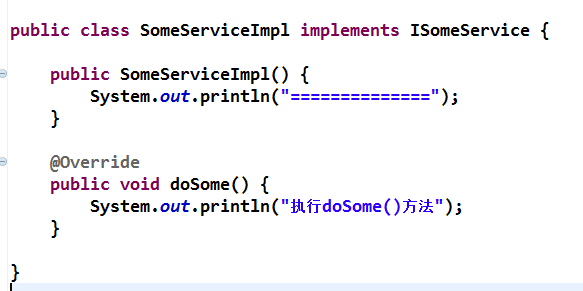


#### 修改测试类

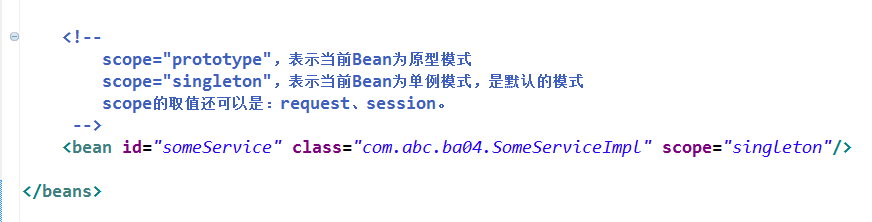


### Bean的作用范围

#### 修改接口实现类



#### 修改配置文件



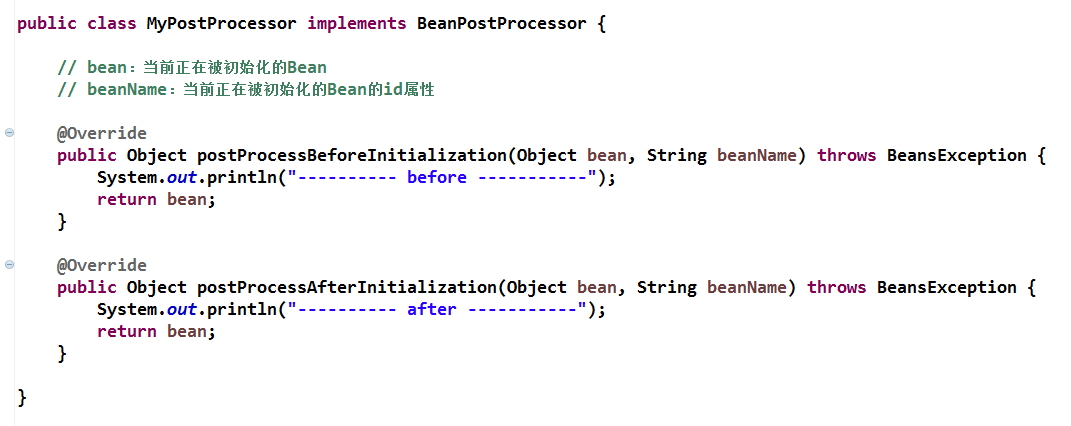
#### 修改测试类



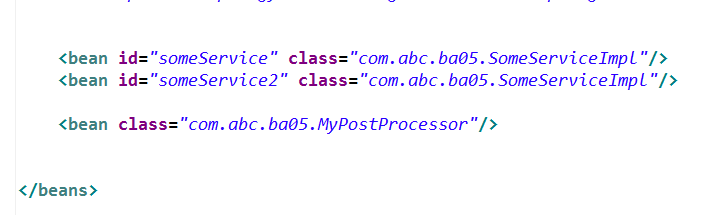
### Bean后处理器

#### 基本原理

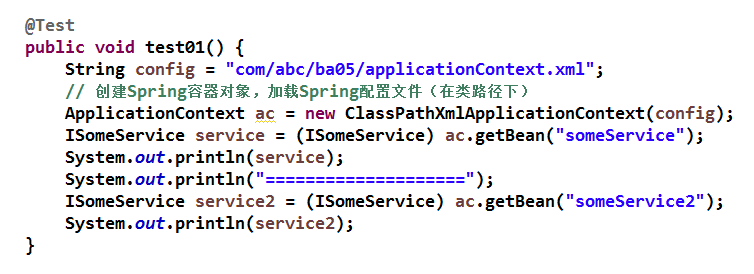
##### 定义Bean后处理器类



##### 修改配置文件



##### 修改测试类



#### Bean后处理器用法示例

### 定制Bean的生命周期始末

### Bean的生命周期

### Bean标签的id与name属性

## 基于XML的DI

DI，依赖注入，即为属性赋值。

### 注入分类

注入分为三类：设值注入、构造注入与实现特定接口注入。

#### 设值注入

调用相关属性的set方法完成的赋值过程，称为设值注入。

#### 构造注入

调用相关的构造器完成的对象创建及初始化过程，称为构造注入。

### 命名空间注入

#### p命名空间设值注入

#### c命名空间构造注入

### 数组与集合属性注入

### 域属性的自动注入

常用的域属性自动注入有两类：

* byName：系统会从容器中查找与域属性同名的bean，将其自动注入
* byType：系统会从容器中查找与域属性类型具有is-a关系的bean，将其自动注入。与域属性类型具有is-a关系的Bean不能多于一个。

### SPEL注入

SPEL，Spring Expression Language，Spring EL表达式。

### 内部Bean注入

### 同类抽象Bean注入

### 异类抽象Bean注入

### 为应用指定多个Spring配置文件

#### 平等关系

#### 包含关系

## 基于注解的DI

### 环境搭建

#### 导入AOP的Jar包

#### 在配置文件添加context约束

#### 在配置文件注册组件扫描器

### DI的基本注解

#### @Component

@Service

@Repository

@Controller

#### @Value

#### 域属性注入注解

@Autowired : byType

@Autowired与@Qualifier：byName

@Resource： JSR250

#### 其它注解

@Scope：

@AfterInit：

@PreDestroy：

#### XML配置与注解的优先级

XML配置的优先级高于注解。

### JavaConfig

使用Java类作为Spring容器，管理Bean的生命周期，及Bean间的依赖关系。

### Spring的JUnit测试

# Spring与AOP

## AOP概述

### AOP简介

AOP，Aspect Orient Programming，面向切面编程。是一种概念，是一种思想，其实现者很多，Spring就是实现者之一。

AOP是OOP的一种补充。OOP是从静态角度考虑程序的结构，而AOP是动态角度考虑程序的运行过程。

Spring用于降低耦合度。IoC用于降低主业务逻辑之间的耦合度；AOP用于降低主业务逻辑与系统级服务（交叉业务逻辑）间的耦合度，将系统级服务在运行时织入到主业务逻辑之中。

Spring的AOP的底层工作原理，是使用动态代理对主业务进行增强。而动态代理使用的是JDK的Proxy与CGLIB两种动态代理。

### AOP相关概念

#### 切面（Aspect）

切面就是交叉业务逻辑，就是要对主业务进行增强的部分。AOP编程中常见的切面有两种：通知、顾问。

#### 织入（Weaving）

将切面切入到主业务逻辑中的过程，称为织入。

#### 连接点（JoinPoint）

可以织入切面的目标类中的目标方法，称为连接点。

#### 切入点（Pointcut）

真正织入切面的目标类的目标方法，称为切入点。切入点一定是连接点。

#### 目标对象（Target）

用于被增强的对象，称为目标对象。

目标类的对象，就是目标对象。

主业务逻辑所在的类的对象，就是目标对象。

#### 通知（Advice）

是一种切面，可以完成简单的织入功能。例如，可以指定织入的时间点。

通知可以指定织入的时间点，切入点指定的是织入的位置。

#### 顾问（Advisor）

是另一种切面，可以完成更复杂的织入功能。其包装了通知。

#### 引入（Introduction）

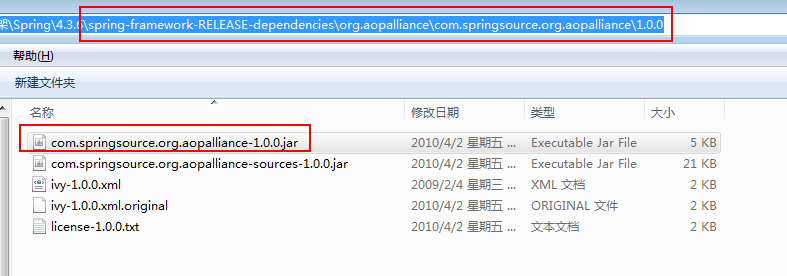
也是一种切面，用于完成的功能是，在不修改目标类的前提下，为目标对象增加新的功能。

## 通知Advice

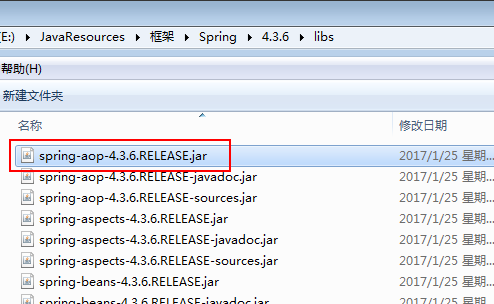
### 编程环境搭建

导入两个Jar包：

AOP联盟：



Spring对AOP的实现：



### 前置通知MethodBeforeAdvice

### 后置通知AfterReturningAdvice

### 环绕通知MethodInterceptor

### 异常通知ThrowsAdvice

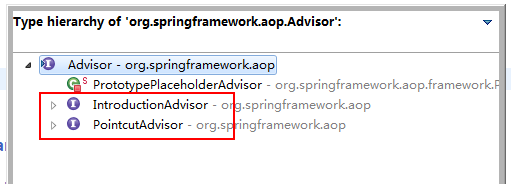
### 为应用指定多个配置文件

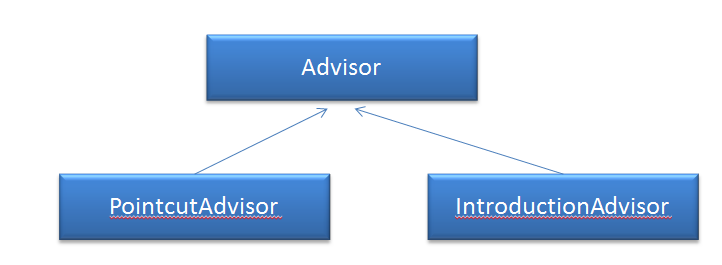
### 无接口时使用CGLIB

### 有接口时使用CGLIB

## 顾问Advisor

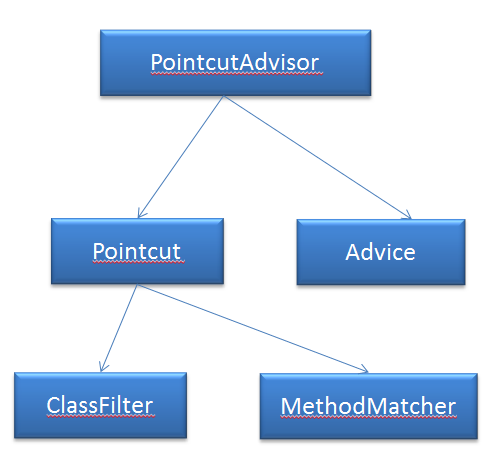
Advisor是另外一种切面，其包装了Advice。





### PointcutAdvisor

切入点顾问。用于指定切入织入的位置，即要将切面织入到哪个目标方法。



#### 自定义切入点顾问

#### 名称匹配方法切入点顾问

#### 正则表达式方法切入点顾问

### IntroductionAdvisor

引入顾问。在不修改目标类的前提下，为目标对象增加新的功能。

## 自动代理生成器

### 默认Advisor自动代理生成器

### Bean名称自动代理生成器

## AspectJ对AOP的实现

### AspectJ概述

#### 什么是AspectJ？

AspectJ是一个面向切面的框架，它扩展了Java语言。AspectJ定义了AOP语法，所以它有一个专门的编译器用来生成遵守Java字节编码规范的Class文件。

#### AspectJ、Spring与AOP的关系

AspectJ是另外一个框架，与Spring没有任何关系。但，AspectJ与Spring都是AOP的实现者。

下面要讲述的是：基于Spring环境下的AspectJ对AOP的实现。

#### AspectJ的通知

前置通知

后置通知

环绕通知

异常通知

最终通知

#### 切入点表达式

用于指定切入点。

execution(

访问权限

返回值类型

全限定性类名（接口名）

方法名(参数列表)

抛出异常

)

execution( \* com.abc.service.\*.\*(..))：表示com.abc.service包下的所有方法都是切入点。

execution( \* com.abc.service.\*.open\*(..))：表示com.abc.service包下的所有以open开头的方法都是切入点。

execution( \* \*.service.\*.\*(..))：表示service包（该service包前只能有一级父包）下的所有方法都是切入点。

com.service.ISomeService.doFirst()是匹配的。

com.abc.service.ISomeService.doFirst()是不匹配的。

execution( \* \*..service.\*.\*(..))：表示service包（该service包前可以有任意级的父包）下的所有方法都是切入点。

com.service.ISomeService.doFirst()是匹配的。

com.abc.service.ISomeService.doFirst()也是匹配的。

### 搭建AspectJ编程环境

#### 导入Jar包

除了Spring基本的Jar包外，还需要四个Jar包：

* AspectJ的核心Jar包
* Spring与AspectJ整合的Jar包
* AOP联盟的Jar包
* Spring对AOP的实现Jar包

#### 引入aop约束

#### 注册AspectJ自动代理生成器

### 基于注解的AspectJ编程

### 基于XML的AspectJ编程

# Spring与DAO

该章节包含两部分内容：Spring的JDBC模板，Spring的事务管理。而这两部分内容，就是对IoC与AOP的典型应用。

* Jdbc模板是对IoC的应用。
* 事务管理是对AOP的应用。

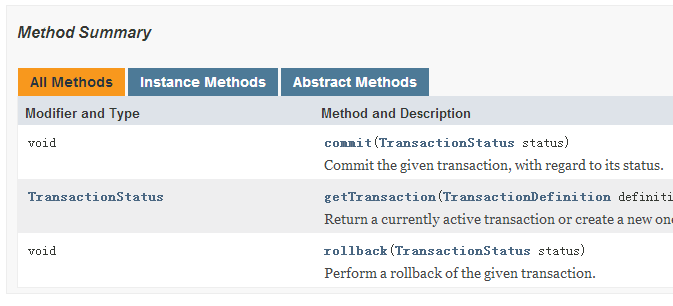
## JDBC模板

## 事务管理

Spring事务管理，主要是要将事务由Dao层提升至Service层。

### Spring事务管理的API

#### 平台事务管理器接口



该接口的功能主要是完成事务的提交与回滚。

##### 常用的两个实现类

* DataSourceTransactionManager

当使用JDBC或MyBatis（iBatis）连接DB，则使用该实现类进行事务管理。

* HibernateTransactionManager

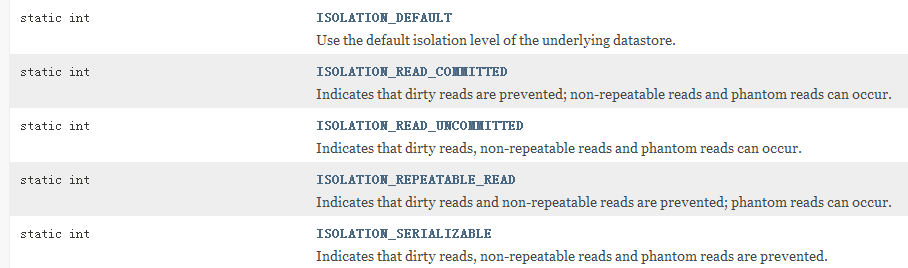
当使用Hibernate连接DB，则使用该实现类进行事务管理。

##### Spring默认的回滚方式

发生运行时异常回滚，发生受查异常提交。该默认回滚方式，程序员通过配置是可以改变的。

#### 事务定义接口

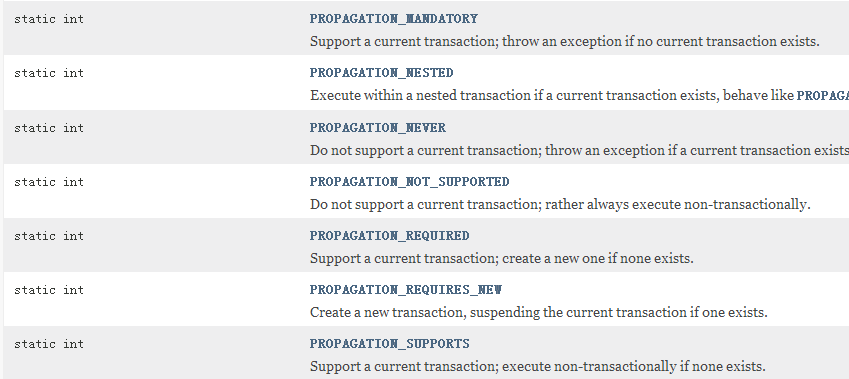
##### 五个事务隔离级别



Mysql默认的事务隔离级别是：可重复读，即4级。

Oracle默认的事务隔离级别是：读已提交，即2级。

##### 七个事务传播行为



### 事物管理举例

购买股票。

#### 使用Spring的事务代理工厂Bean管理事务

#### 使用Spring的事务注解管理事务

#### 使用AspectJ管理事务（重点）

# Spring与MyBatis

MyBatis与Spring整合，其主要任务是将MyBatis的SqlSessionFactory交由Spring容器来管理。

# Spring与Web