1.下载jar包地址：

<http://repo.springsource.org/libs-release-local/org/springframework/spring/>

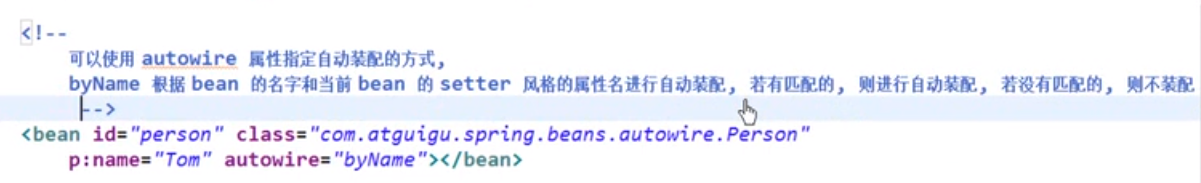
2.下载缺少的jar包

<http://www.java2s.com/Code/Jar/c/Catalogc.htm>

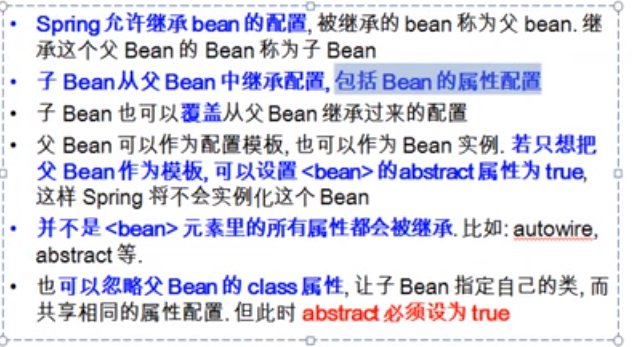
## IOC&DI概述

### 配置bean

* + 1. 形式：基于xml文件、注解
    2. Bean的配置方式
       1. 通过全类名（反射）
       2. 通过工厂方法（静态工厂方法&实例工厂方法）
       3. FactoryBean
    3. Ioc容器：BeanFactory和ApplicationContext（使用）
    4. 依赖注入的方式：属性（set和get）注入、构造器注入 、工厂方法注入（不用）
    5. 自动装配
       1. 基于xml的装配，有byname（类中的属性会在applicationContext.xml中找bean id）和bytype （找相同类型，如果类型有重复就不行）



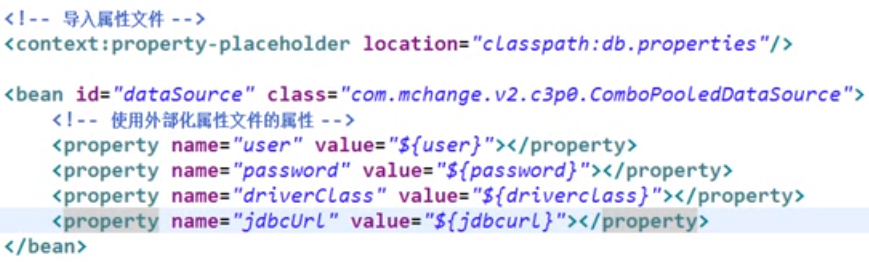
* + - 1. 基于注解的
    1. Bean之间的关系：继承和依赖
       1. 继承bean配置（不光继承属性名，还继承属性值）



* + - 1. 依赖bean配置

通过depend-on属性设定bean的前置依赖bean，会在本实例实例化之前创建好，如果依赖多个bean，则通过逗号，空格等方式配置

* + 1. Bean的作用域
       1. SingleTon，默认的作用域是单例的，整个容器生命周期都只产生一个bean。而且创建IOc容器的时候会自动创建配置的bean。
       2. 可以配置成propotype，每次都返回一个新的bean。容器初始化时不创建bean实例，每次请求的时候才创建。
       3. 还有request和session
    2. 使用外部文件（加载db.properties）



* + 1. SpEL
       1. Spring表达式语言，可以运行时查询
       2. 语法类似于EL，用#{ }作为界定符
    2. IOC容器中bean的生命周期
       1. 过程：

通过构造器或工厂方法创建bean实例

为bean的属性设置值和对其他bean的引用

调用bean的初始化方法

可以使用bean

容器关闭，调用bean销毁方法

* + - 1. 在bean的声明里，设置init-method和destory-method可以设置初始化和销毁方法。（销毁方法可以写在类里）

### 基于注解的配置

* + 1. 组件扫描
       1. 在classpath下自动扫描特定注解的组件
       2. 特定组件包括：

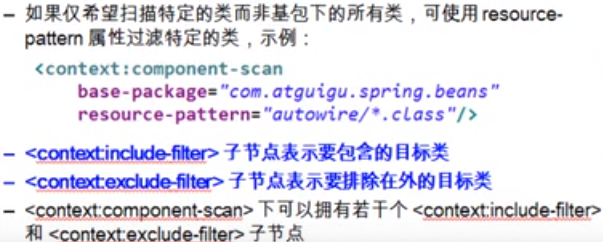
@Component 基本注解

@Repository 持久层

@Service 业务层

@Controller 表现层

* + - 1. 对于扫描到的组件，Spring默认使用非限定类名，第一个字母小写。
      2. 扫描组件：配置<context：component-scan>



* + - 1. 组件装配

有三种，@Autowired(常用) @Resource @Inject

### 知识点

* + 1. 配置成bean的类中必须要有一个无参的构造方法，因为Ioc通过反射的方式来创建bean。
    2. Bean内部也可以配置bean，内部bean不能在外面引用
    3. 在applicationContext.xml中为list赋值，可以在propety标签里使用list标签。set标签也是一样。map标签内部通过 entry标签来设置，entry标签里又key和vaule-ref
    4. 配置Properties属性值

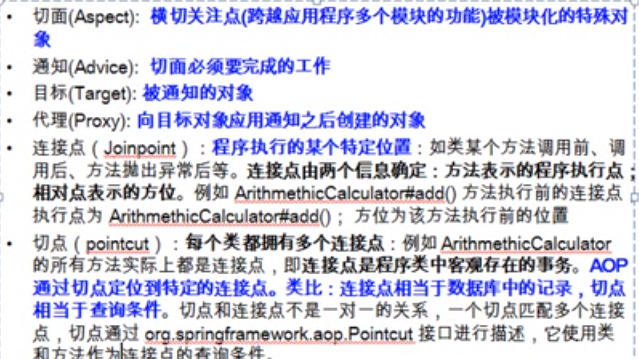


## AOP

解决代码混乱和代码分散问题

### 概念

* + 1. 切面（Aspect），通知（Advice），目标（target），代理（proxy），连接点（joinpoint），切点（cutpoint）



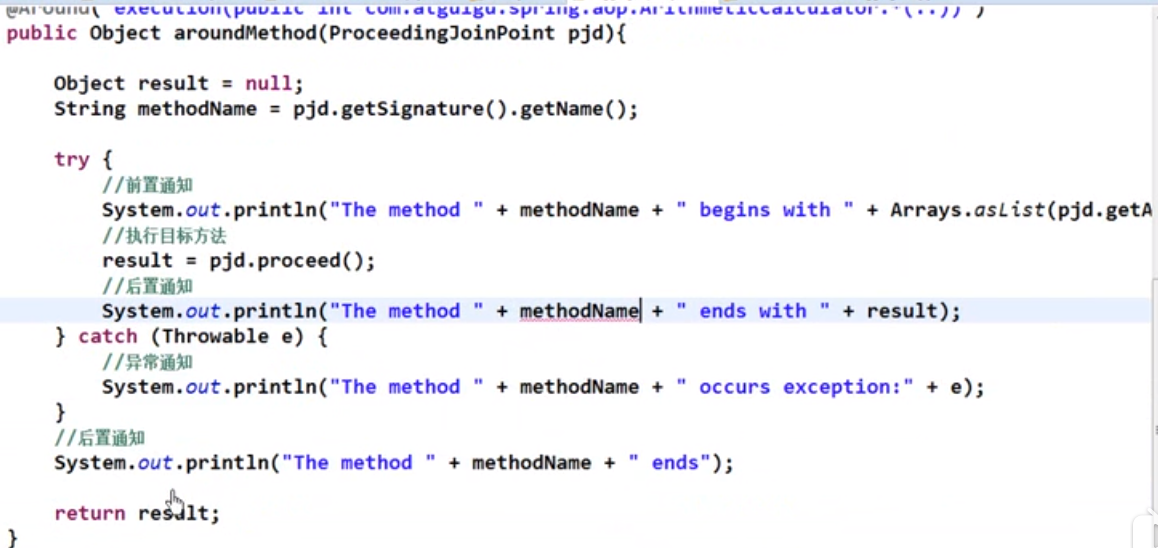
* + - 1. 通知的种类：

@Before @After @AtferReturning @AfterThowing @Around

* + 1. Aspect Aop框架
       - 1. 基于注解的方式：
       1. 定义一个切面类，写上对应的注解（@component和@Aspect），在xml文件中添上<aop:aspect-autoproxy></aop:aspect-autoproxy>
       2. 在类中声明各种方法，加上对应的通知种类（例如：@After）,然后再括号里配置参数 例如：@Before(“excution(方法的完成名称)”)

### 知识点

1. 各种通知：
   1. 后置通知是在连接点完成之后执行的，即连接点返回结果或者抛出异常的时候。
   2. @AtferReturning是方法正常结束之后才执行，所以是可以访问到方法的返回值的
   3. 每一个通知方法参数列表里使用joinpoint 可以获得方法的参数和名称。而@Around里面使用ProceedingJoinPoint 比如：



* 1. 可以在切面类上使用@Order（）来指定切面的优先级，括号里填数字，1最高
  2. 可以定义一个方法，注解上@PointCut来声明切入点表达式，然后通知的直接使用方法名来引用切入点

