写此笔记为了让以后脑残的自己再重新接触这一部分的时候不至于像当初的jsp似的，学多少忘多少，最后只能重新开始，而且以前写的笔记又烂到不行，虽然很有条理，但自己其实按照步骤都看不懂，真是蠢，写此文档时，切记稍安勿燥，能把自己当多蠢当多蠢，假设自己什么都不会写每一步

1. 国际惯例，编程语言第一步（helloWorld）

IOC（DI）：控制反转（依赖注入）：懂不懂的现在先记着

1. 首先，要先把spring的这个工具导入到eclipse当中，自己去官网下载。



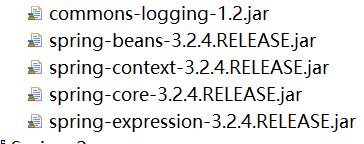
在help那有个installnewSoftWare，然后add，archive，自己搞一搞

1. 要想在eclipse上用spring实现helloworld，需要导入5个基础的jar包（jar包还是去官网下载）

方法是，在项目下创建lib文件夹存放这些jar包（这些jar包是必须的，千万注意那个commons-logging）



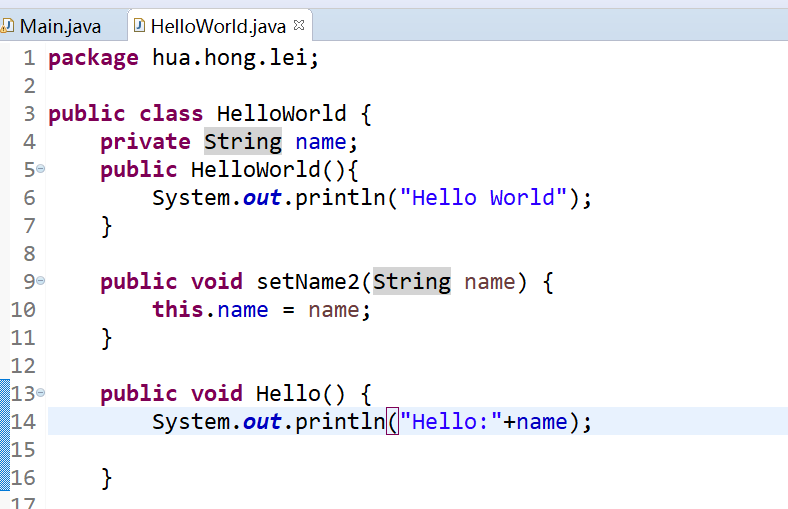
在官网上下载了这个解压即可



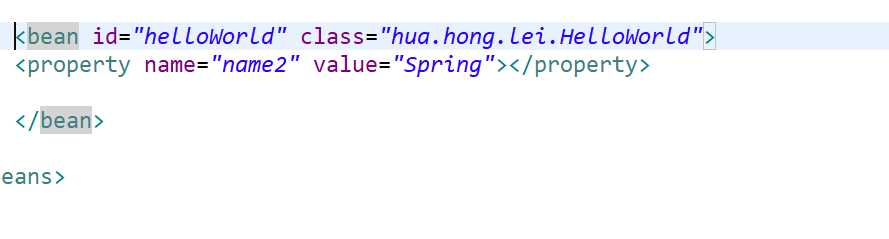
好吧，导入完之后别让了build path 然后add什么玩意就行

然后这里的helloworld，因为单写spring的helloworld的话只需要创建一个简单的java工程即可

3然后写一个helloWorld的javaBean（javabean含义自行百度），

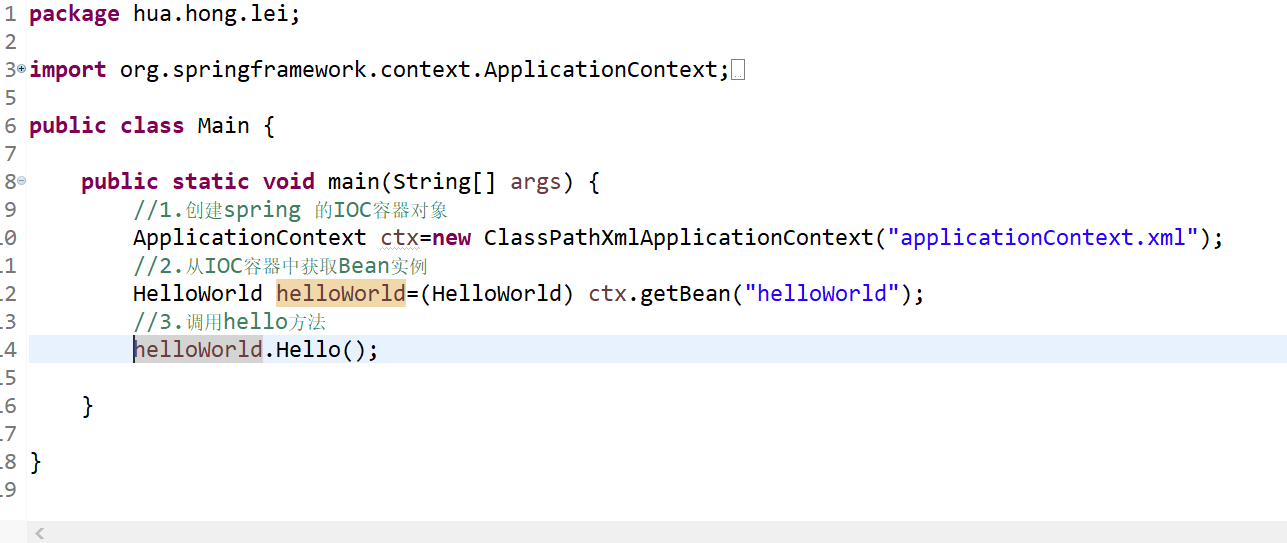


1. 在src下写一个spring的xml文件，可以看出class就是你那个全类名，id是来唯一标识这个bean，property字标签是为了给javabean里面的属性赋值



希望你看过的课本还没忘，这很显然是装配javaBean的三种方式的其中一种（xml文件的装配）

5.再写一个测试类，里面照着写



6.如果你还有点救的话，这个时候你就会挨着注释，看看每一步都有什么用

这时候你会发现创建IOC容器对象那一步其实是创建了一个对象，且给那个对象的name属性赋给xml文件中的值：Spring。

然后下面就是理所应当的从IOC容器里提取出实例，然后这个对象调用它的一个方法。

1. IOC和DI

IOC：其思想是反转资源获取的方向。传统的资源查找方式要求组件向容器发起查找资源作为回应，容器适时的返回资源，而应用了IOC之后，则是容器主动地将资源推送给它所管理的组件，组件所要做的仅是选择一种合适的方式来接受资源，这种行为也被称为查找的被动形式。

DI：IOC的另一种表述方式：即组件以一些预定义好的方式（例如：setter方法）接受来自如容器的资源注入，相对于IOC而言，这种表述更为直接

三、配置bean

1.配置形式：

①基于XML文件的方式；

②基于注解的方式

a)组件扫描：Spring能够从classpath下自动扫描，侦测和实例化具有特定注解的组件

特定组件包括

-@Component：基本注解，标识了一个受Spring管理的组件

-@Repository：标识持久层组件

-@Service：标识服务层（业务层）组件

-@Controller：标识表现层组件

对于扫描到的组件，Spring有默认的命名策略：使用非限定类名，第一个字母小写，也可以在注解中通过value属性值标识组件的名称

b).当在组件类上使用了特定的注解之后，还需要在Spring、

的配置文件中声明<context:component-scan>

-base-package属性指定一个需要扫描的基类包、Spring容器将会扫描这个基类包及其子包中的所有类

-当里面需要扫描叫多个包时。可以使用逗号分量

-如果仅希望扫描特定的类而非基类包下的所有类，可使用resource-pattern属性过滤特定的类

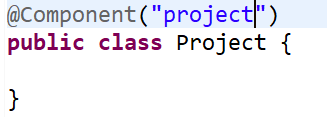
-<context:include-filter>子节点表示要包含的目标类

-<context:exclude-filter>子节点表示要排除在外的目标类

-<context:component-scan>下可以拥有若干个<context:include-filter>和<context:exclude-filter>子节点

c).实例

-先在Bean上写上注解，注解后面有value属性，可以表明Bean的id，value属性是默认的，可以直接在括号里写，而且，一般不用写，默认为类名第一个字母小写，不在接口上写，在实现类上写。。。例如Dao接口和Daoimpl，就需要写在Daoimpl上



-在xml文件里写上配置，自动扫描包了。。这样包中的类上的注解才有用

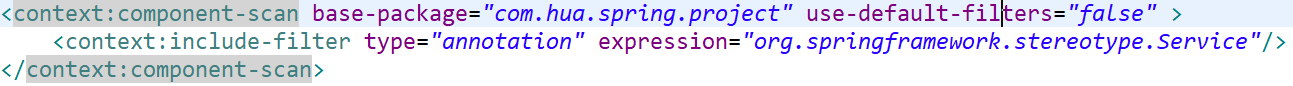


-resource-pattern可以用来过滤特定的类，下面例子的意思是com.hua.spring.project的子包com.hua.spring.project.service下的所有的类

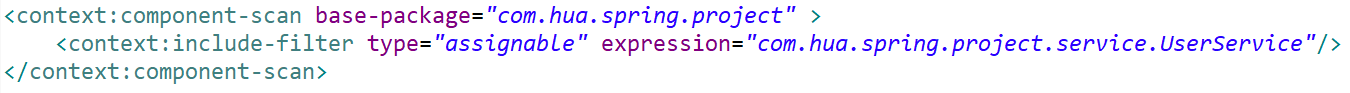


-<context:include-filter>子节点表示要包含的目标类，type属性一般用annotation和assignable两种，要想这个子节点有用，得用use-default-filters属性，得将这个属性设置为false，要么默认是扫描所有的注解

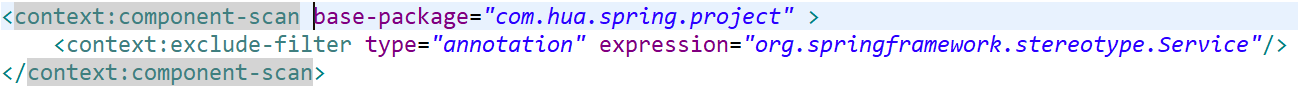
-当type为annotation时，expression时写注解的全类名



-当type为assignable时，expression时写类的全类名



-<context:exclude-filter>子节点表示要排除在外的目标类



d）.当一个**Bean里面装配其他的bean时**

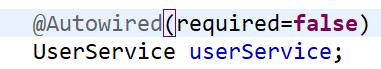
**<context:component-scan>元素还会自动注册AutowiredAnnotationBeanPostProcessor实例，该实例可以自动装配具有@Autowired和@Resource、@Inject注解的属性，下面学习可以再一个Bean 里面装配另一个Bean的Autowired注解的方式**

**e)**[**.@Autowired**](mailto:.@Autowired)**注解自动装配具有兼容类型的单个Bean属性**

**-构造器，普通字段（即使是非public），一切具有参数的方法都可以应用@Autowired注解**

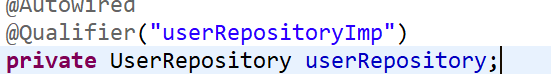
****

**-默认情况下，所有使用@Autowired注解的属性都需要被设置，当Spring找不到匹配的Bean装配属性时，会抛出异常，若某一属性不被设置，可以设置@Autowired注解的required属性为false，就是如果IOC容器中没有这一个bean而且还不想让报错的话，可以如下设置**

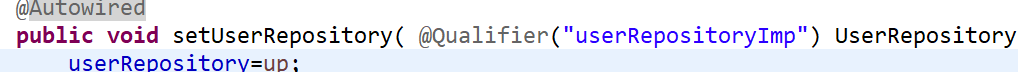
****

**-默认情况下，当IOC容器里存在多个类型兼容的Bean时，通过类型的自动装配将无法工作，此时可以再@Qualifier注解里提供Bean的名称，Spring允许对方法的入参标注@Qualifier已指定注入Bean的名称。还可以将某一兼容Bean的名称设置成与当前属性相匹配的模式**

****

****

**--在入参前面加注解**

****

**-@Autowired注解也可以应用在数组类型上，此时Spring将会把所有匹配的Bean进行自动装配**

**-@Autowired注解也可以应用在集合属性上，此时Spring读取该集合的类型信息，然后自动装配所有与之兼容的Bean**

[**-@Autowired注解用在java.util.Map**](mailto:-@Autowired注解用在java.util.Map)**上时，若该Map的键值为String，那么Spring将自动装配与之Map值类型兼容的Bean，此时Bean的名称作为键值**

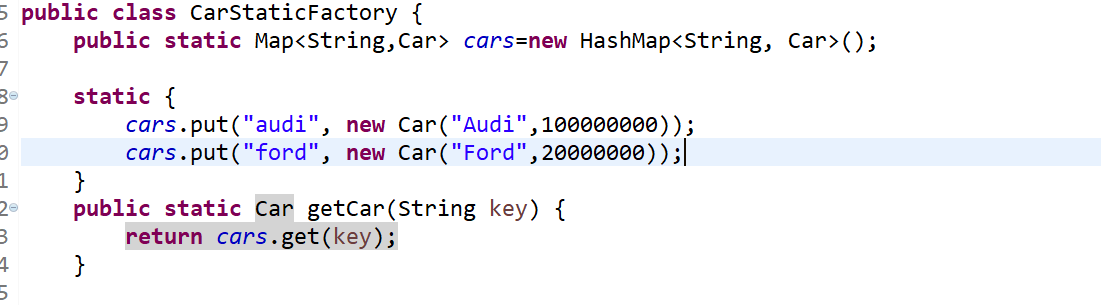
2.Bean的配置方式：

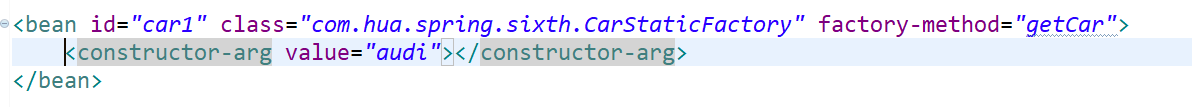
1. .通过类名（反射）:注意，因为是反射所以要有无参构造器
2. .通过工厂方法（静态工厂方法&实例工厂方法）

a).静态工厂方法

-调用静态工厂方法创建Bean 是将对象创建的过程封装到静态方法中，当客户端需要对象时，只需要简单地调用静态方法，而不用关心创建对象的细节

-要声明通过静态方法创建的Bean，需要在Bean的class属性里指定拥有该工厂的方法的类，同时在factory-method属性里指定工厂方法的名称，最后，使用<constructor-arg>元素为该方法传递方法参数





b)实例工厂方法

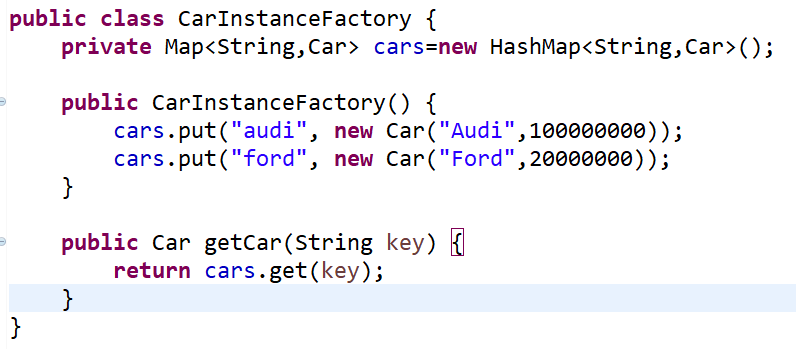
-实例工厂方法，将对象的创建过程封装到另外一个对象实例的方法里，当客户端需要请求对象时，只需要简单的调用该实例方法而不需要关心对象的创建细节

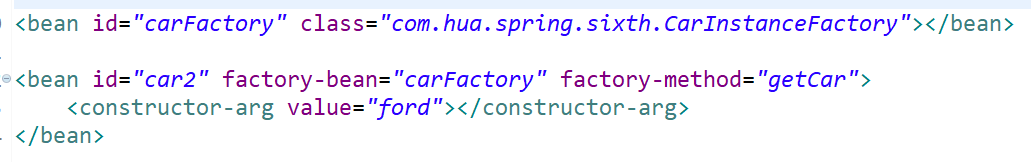
-要声明通过实例工厂方法创建的Bean

-在bean 的factory-bean属性里指定拥有该工厂方法的Bean

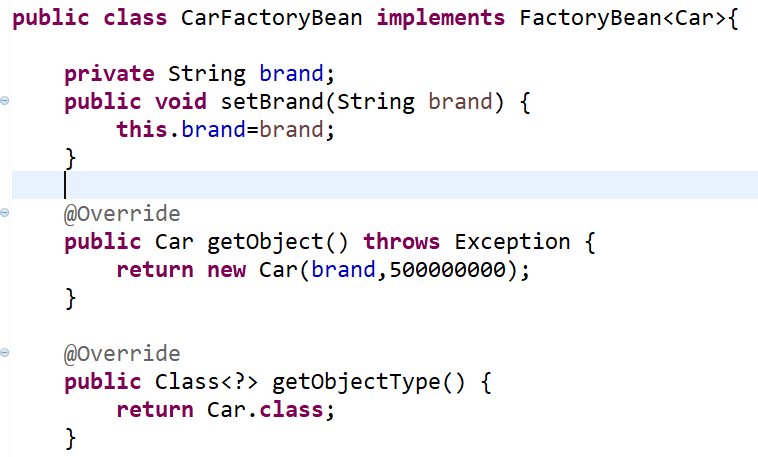
-在factory-method属性里指定该工厂方法的名称

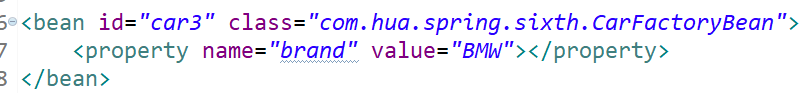
-使用constructor-arg元素为工厂方法传递方法参数





1. FactoryBean





3．IOC容器：BeanFactory&ApplicationContext概述

1. ApplicationContext：提供了更多的高级特性，是BeanFactory的子接口
2. BeanFactory：是Spring框架的基础设施，面向Spring本身
3. ApplicationContext面向使用Spring框架的开发者，几乎所有的应用场合都直接使用ApplicationContext而非底层的BeanFactory
4. ApplicationContext有两个主要实现类
   1. ClassPathXmlApplicationContext
   2. FileSystemXmlApplicationContext
   3. 接口ConfigurableApplicationContext扩展于ApplcationContext，新增加两个主要方法：refresh（）和close（），让ApplicationContext具有启动、刷新和关闭上下文的能力
5. ApplicationContext在初始化上下文就实例化所有单例的Bean
6. 从IOC容器中获取Bean

getBean（……）：有很多种重载的版本，具体看文档，有String类型，还有class类型

注意用class类型的重载版本时，不用强转。



5.依赖注入的方式：属性注入；构造器注入；工厂方法注入（知道这回事就行）

①属性注入

a)属性注入即通过setter方法注入Bean属性值或依赖的对象

b)属性注入使用<property>元素，使用name属性指定Bean的属性名称，value属性或<value>子节点执行属性值

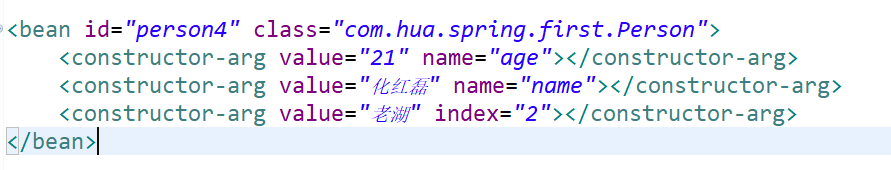
c)属性注入是实际应用中最常用的注入方式

②构造器注入

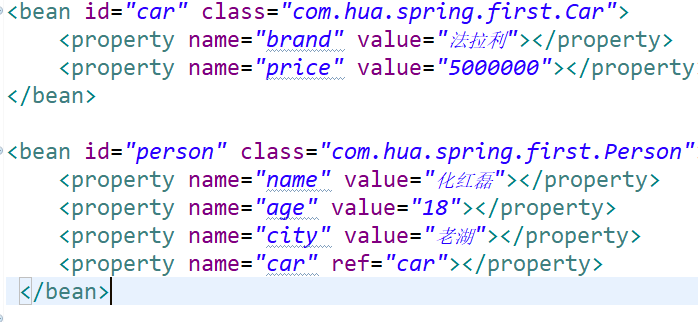
a)通过构造器注入Bean的属性或依赖的对象，它保证了Bean实例在实例化就可以使用

b)构造器注入在<constructor-arg>元素里声明属性，直接按照构造器的值的顺序写

c)使用构造器注入属性值可以指定参数的位置（index）和参数的类型（type），以区分重载的构造器



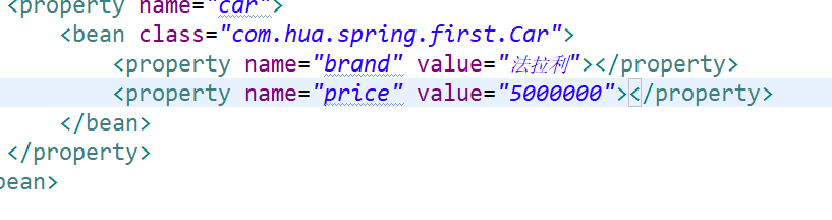
1. 注入属性值细节
   1. .引用数据类型的注入：在property 和constructor-arg里面都有一个ref属性，用ref属性来指向引用的那个值就行了



* 1. Null属性的赋值，用专门的<null/>来赋值，直接写null会报错

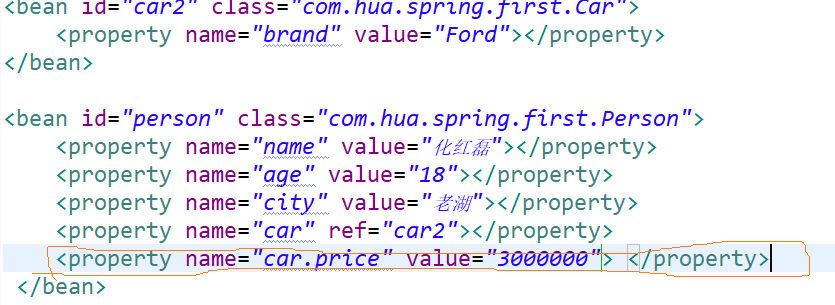


* 1. 内部bean：内部bean只能在内部使用，不能再外部使用，所以声明内部bean 时可以没有id属性



* 1. 级联属性赋值，注意，属性需要先初始化后才能为级联属性赋值，否则有异常

即，先得有car之后才能为car.price赋值



* 1. 有特殊字符的，比如说<>这个就是xml文件里的特殊字符，要这样写

<property name=*"voice"* >

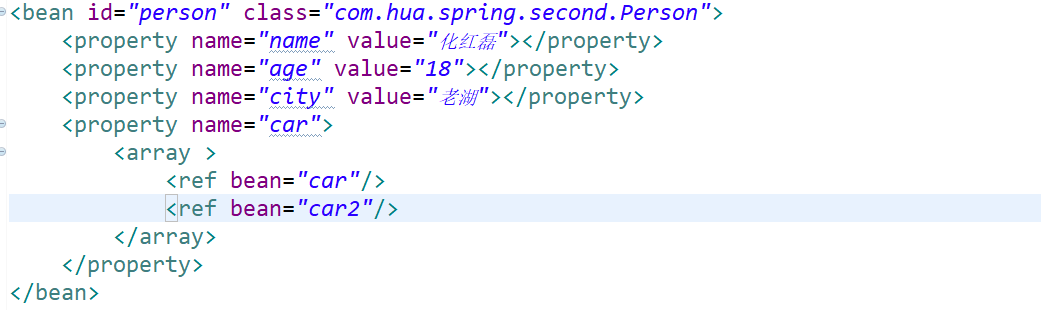
<value><![CDATA[<MiaoMiao>]]></value>

</property>

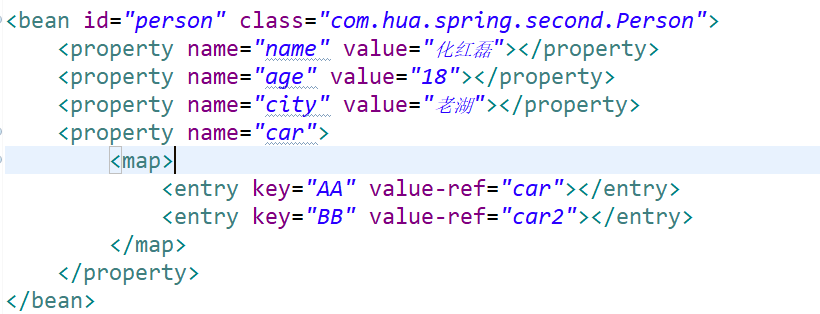
* 1. 配置java.util.List类型的属性，需要制定<list>标签，在标签里包含一些元素，这些标签可以通过<value>制定简单的常量值，通过<ref>指定对其他Bean的引用，通过<bean>指定内置Bean定义，通过<null/>指定空元素，甚至可以内嵌其他集合



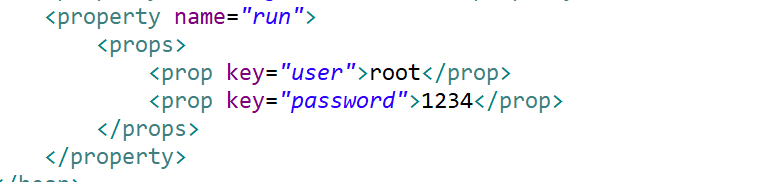
* 1. 数组的定义和List差不多



* 1. 配置java.util.Map

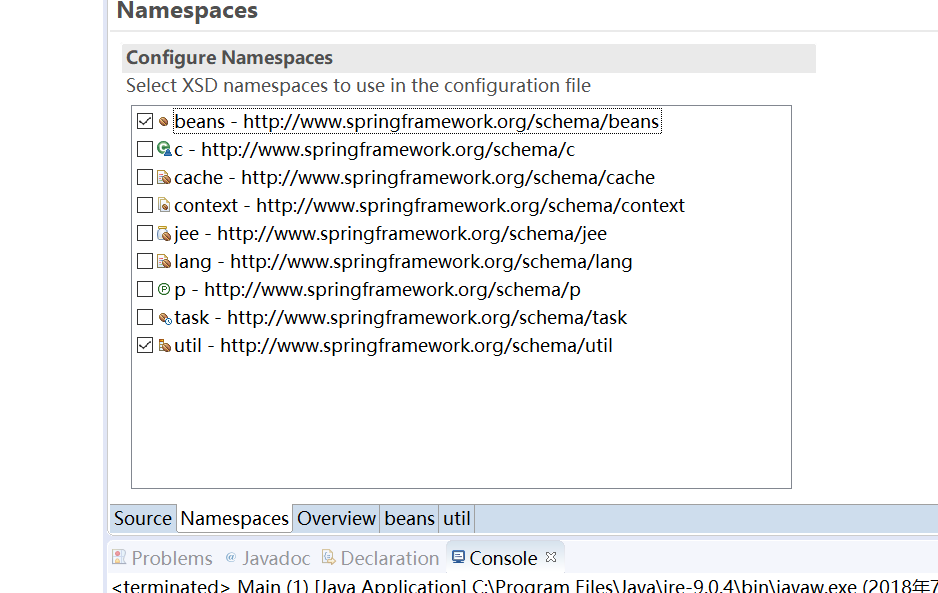


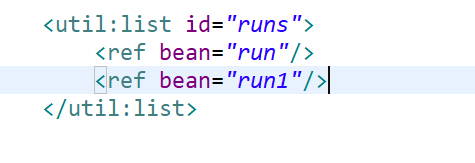
* 1. 配置java.util.Properties



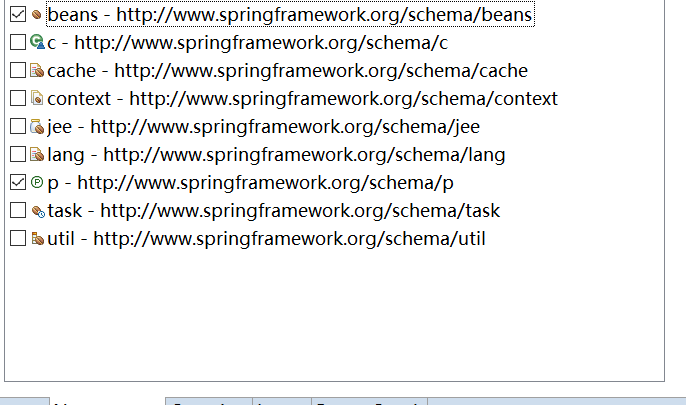
* 1. -使用utility schema定义集合：使用基本的集合标签定义集合时，不能将集合作为独立的Bean定义，导致其他Bean无法引用该集合，所以无法在不同Bean之间共享集合，，，可以使用util schema里的集合标签定义独立的集合Bean，需要注意的是，必须在<beans>根元素里添加util schema定义

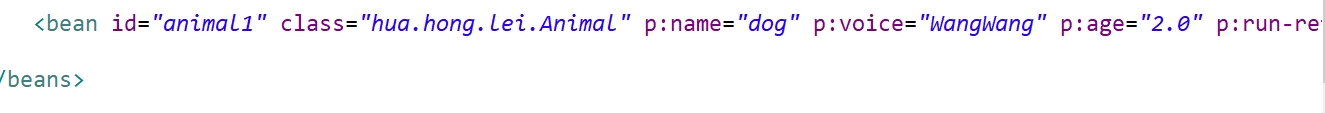
-首先，引入新的命名空间，然后就可以使用了





* 1. 使用p空间命名：为了简化xml文件的配置，越来越多的xml文件采用属性而非子元素配置信息。（话不多说，看看例子就懂了）

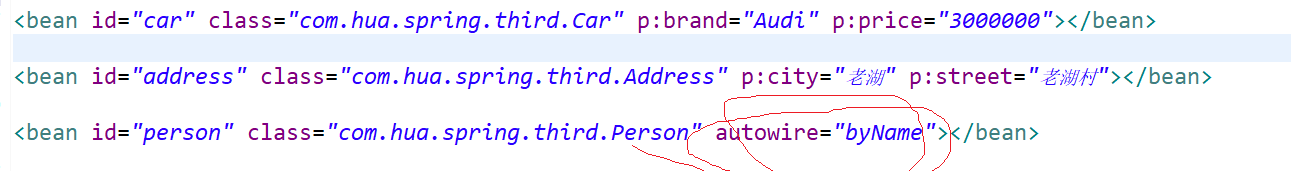




1. 自动装配
   1. 可以使用autowire属性指定自动装配的方式，

byName：根据bean的名字和当前bean的setter风格的属性名进行自动装配，若有匹配的，则进行自动装配，若没有匹配的，则不装配

byType：根据bean的类型和当前bean的属性的类型进行自动装配，若IOC容器中有1个以上的类型匹配的bean，则抛异常



* 1. 自动装配的缺点：

-在Bean配置文件里设置autowire属性进行自动装配将会装配Bean的所有属性，然而，若只希望装配个别属性时，autowire属性就不够灵活了

-autowire属性要么根据类型自动装配，要么根据名称自动装配，不能两者兼而有之

- 一般情况下，在实际的项目中很少使用自动装配功能，因为和自动装配功能所带来的好处比起来，明确清晰的配置文档更有说服力一些。

-但在整合第三方框架时，自动装配会带来方便

7．bean之间的关系：继承；依赖

①继承Bean配置

a)Spring允许继承bean的配置，被继承的bean称为父bean，继承这个父Bean的Bean称为子Bean

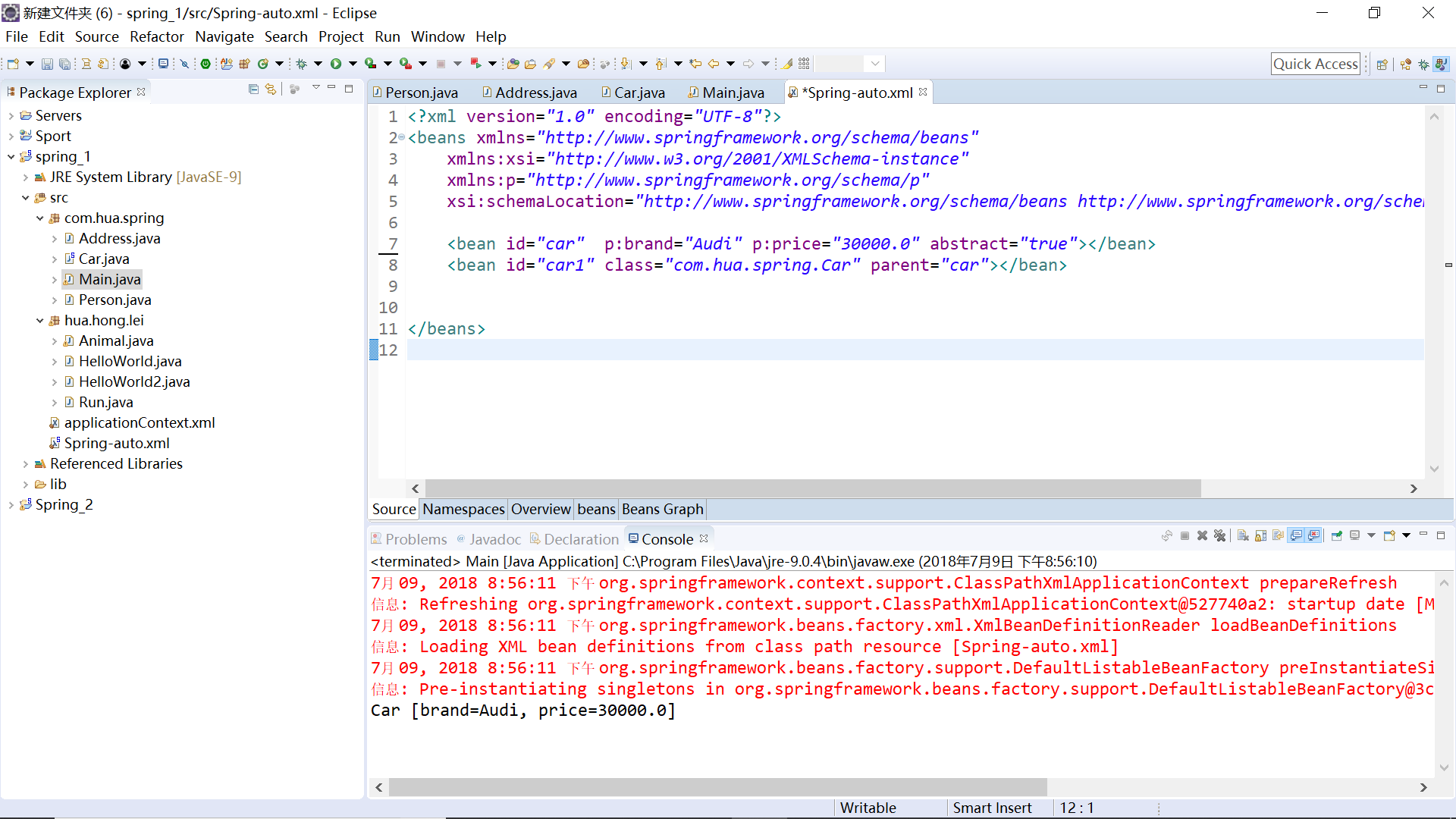
b)子Bean从父Bean中继承配置，包括Bean的属性配置

c)子Bean也可以覆盖从父Bean继承过来的配置

d) 父Bean可以作为配置模板，也可以作为Bean的实例，若只想把父Bean作为模板，可以设置<bean>的abstract属性为true，这样spring将不会实例化这个Bean

e) 并不是<bean>元素里的所有属性都会被继承，比如：autowire，abstract等

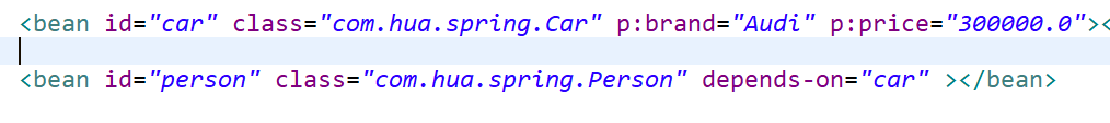
f)也可以忽略父Bean的class属性，让Bean指定自己的类，而共享相同的属性配置，但此时abstract必须设为true



* 1. 依赖Bean配置

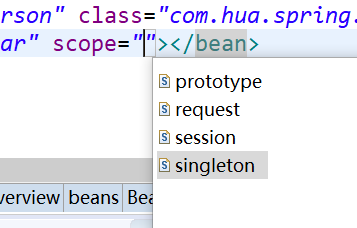
A）.Spring允许用户通过depends-on属性设定Bean前置依赖的Bean，前置依赖的Bean会在本Bean实例化之前创建好，如果没创建好就会报错

* + 1. 如果前置依赖于多个Bean，则可以通过逗号，空格方式配置Bean的名称



**8.bean的作用域：singleton；prototype；WEB环境作用域**

* 1. **配置方式**



singleton：bean的默认作用域为singleton，单例的，容器初始化时创建bean容器，每次向IOC容器中获取bean，都是相同的bean。在整个容器的声明周期内只创建一个bean。

**prototype：原型的，即每次向IOC容器获取bean时，都是不同的bean。在容器初始化时不创建bean 的实例，而在每次请求时都创建一个新的bean实例，并返回**

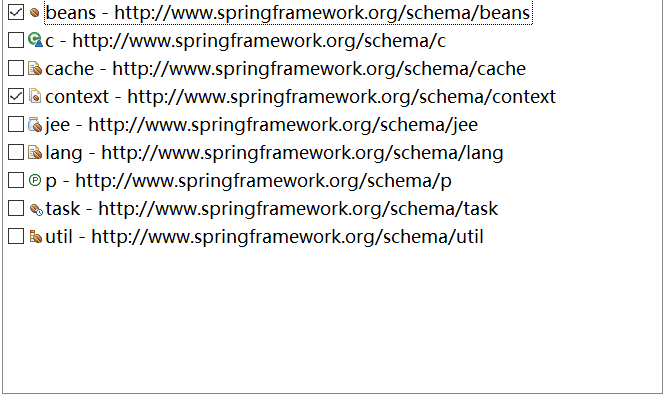
**WEB环境作用域：request、session**

9.使用外部属性文件

在配置文件里配置Bean时，有时需要在Bean的配置里混入系统部署的细节信息（例如：文件路径，数据源配置信息等）。而这些部署细节实际上需要和Bean配置相分离

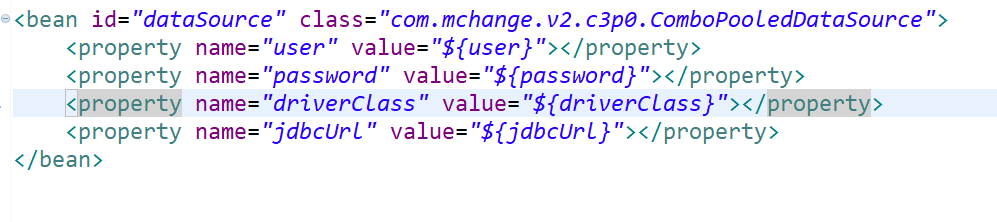
Spring提供了一个PropertyPlaceholderConfigurer的BeanFactory后置处理器，这个处理器允许用户将Bean配置的部分内容外移到属性文件中，可以再Bean配置文件里使用形式为${var}的变量，PropertyPlaceholderConfigurer从属性文件里加载属性，并使用这些属性来替换变量

1. 不用外部属性文件配置mysql数据库连接的版本
2. 用外部属性文件配置mysql数据库连接的版本
3. 引入context命名空间



1. 然后，配置context：property-placeholder



1. 

10.SpEL

①．简介

-Spring表达式语言（简称SpEL）：是一个支持运行时查询和操作对象图的强大的表达式语言

-语法类似于EL：SpEL使用#{..}作为定界符，所有在大括号中的字符都将被认为是 SpEL

-SpEL为bean的属性进行动态赋值提供了便利

②SpEL具体功能

A）.引用字面值（普通的int String等值）引用String的值，要用’’括起来

B）.引用Bean、属性和方法

即可以引用其他对象、引用其他对象的属性、调用其他方法，还可以链式操作

C）.引用一个类中的某个静态变量



11.IOC容器中Bean的生命周期

①SpringIOC容器可以管理Bean的生命周期，Spring允许在Bean生命周期的特定点执行定制的任务。

②SpringIOC容器对Bean的生命周期进行管理的过程：

A).通过构造器或工厂方法创建Bean实例

B).为Bean的属性设置值和对其他Bean 的引用

C).调用Bean的初始化方法

D).Bean可以使用了

E).当容器关闭时，调用Bean 的销毁方法

1. 在Bean的声明里设置



④ 更细粒度的定制Bean声明周期（创建Bean的后置处理器）

A).通过构造器或工厂方法创建Bean实例

B).为Bean的属性设置值和对其他Bean 的引用

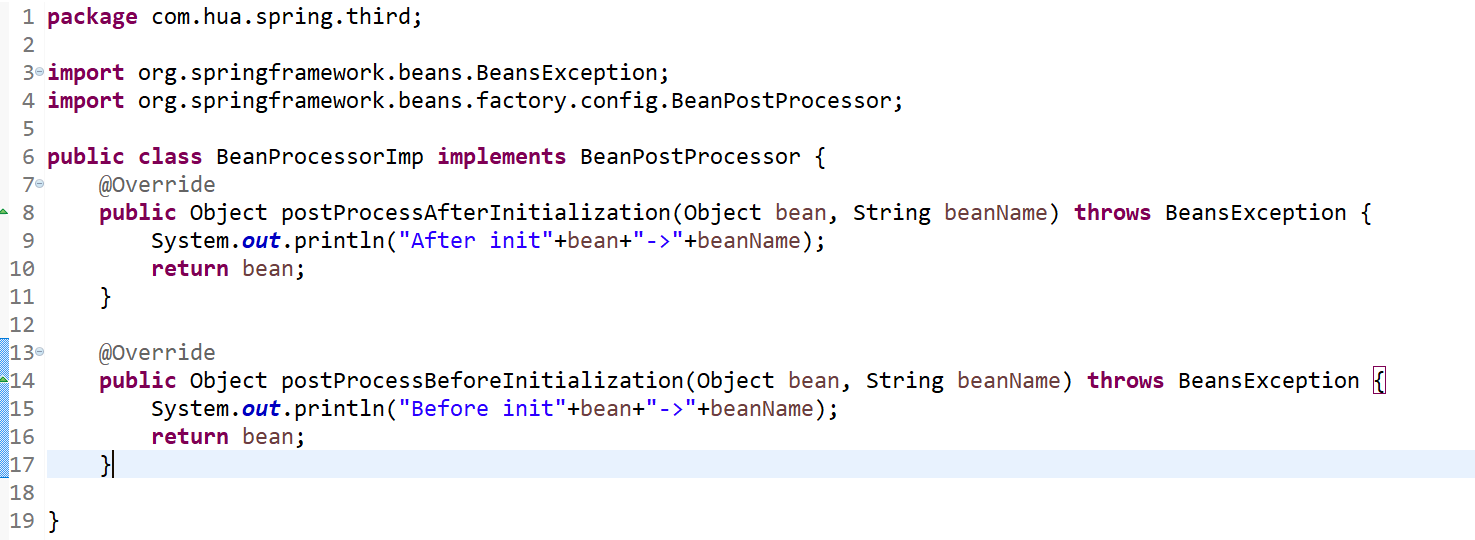
C).postProcessBeforeInitialization方法

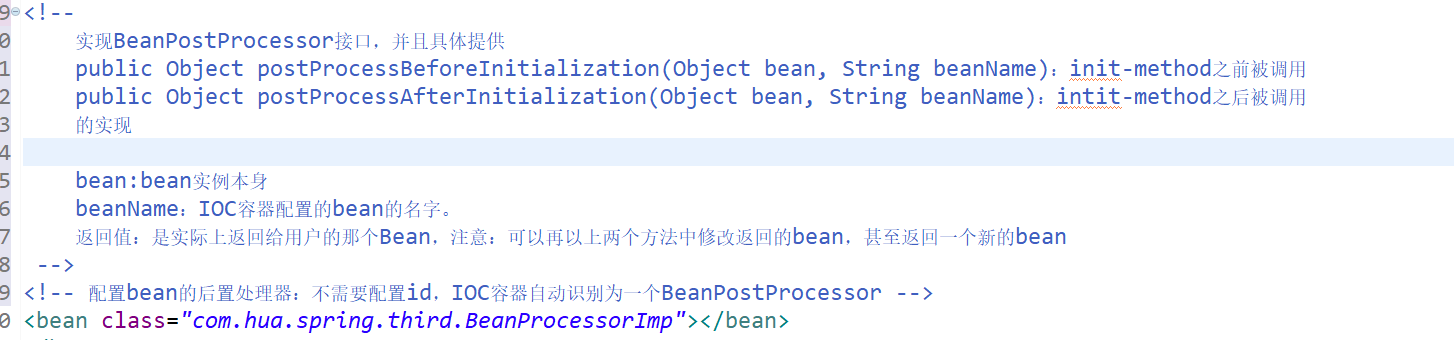
D).调用Bean的初始化方法

E). postProcessAfterInitialization方法

F).Bean可以使用了

G).当容器关闭时，调用Bean 的销毁方法





12

-Spring4.x新特性：泛型依赖注入