1. 设置属性-给属性赋值

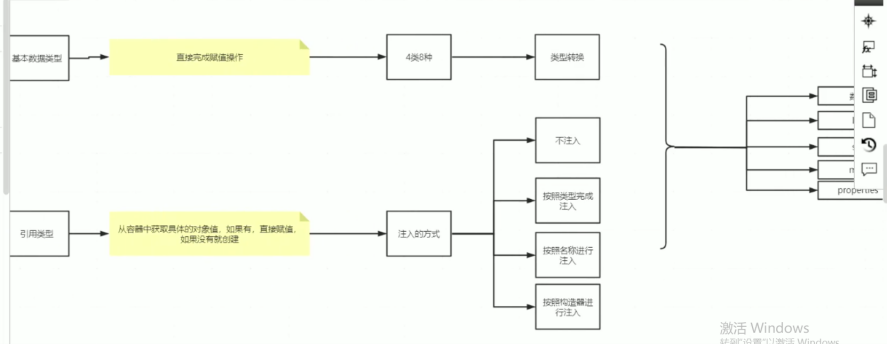
设置属性的时候，就会确定属性是普通对象还是代理对象。

1. 属性
2. 属性类型

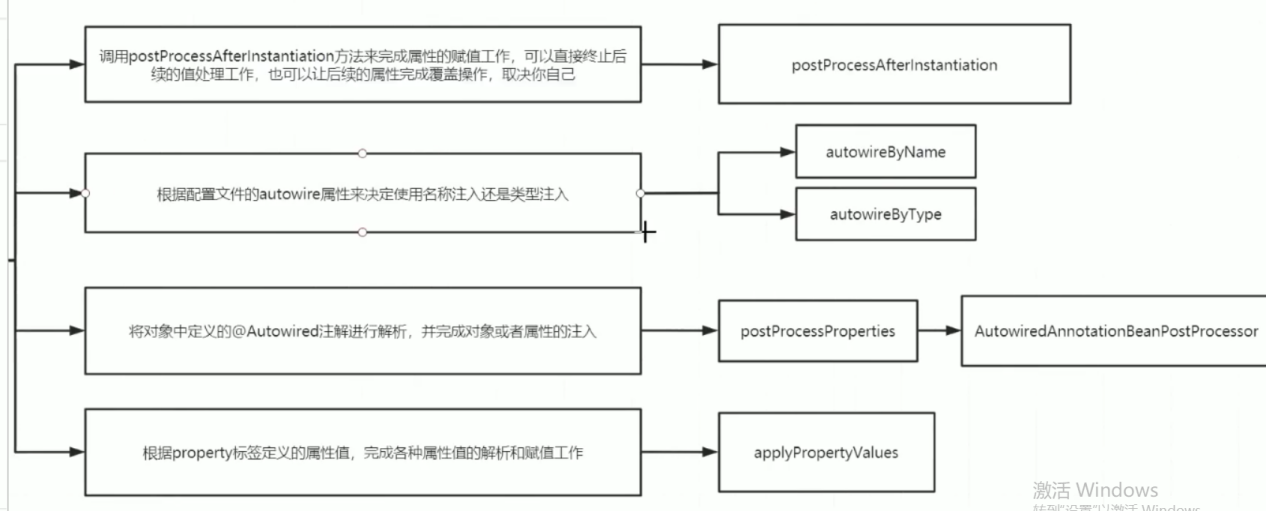
基本数据类型、引用类型

1. 引用类型注入方式

不注入、类型注入、名称注入、构造器注入



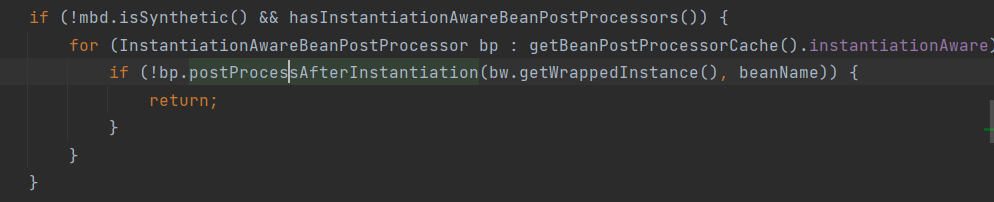
1. populateBean填充属性（属性：配置文件中<property></property>标签）



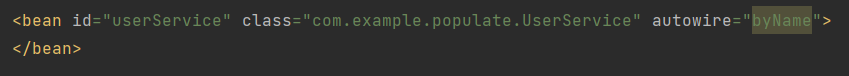
应对不同的场景，如果写的是配置文件则按照第四步进行处理；如果是配置文件与注解混用的场景，也需要对这些注解进行识别，按照二三两步处理。

1. 如果有自定义扩展的InstantiationAwareBeanPostProcessor

则可以使用它的postProcessAfterInstantiation方法，给类中属性赋值（无关配置文件），如果返回false则不用往下处理；主要是后面赋值太复杂了，简化赋值操作。



1. 获取所有标签属性PropertyValues（在BeanDefinition中），即标签<property></property>
2. 如果配置文件有写auto-wired，获取类中所有非简单、有set方法的属性值，存储到PropertyValues中（类中：所有非简单并且有set方法的属性）



* autowireByName通过名字查找对应的属性对象

1. 扫描类中所有属性，对于有set方法并且非简单的属性，按名字从容器中获取属性值，存储到PropertyValues中。
2. 存储依赖关系（可能多次创建当前对象，对于依赖的属性可以直接从缓存中取）

* autowireByType通过类型存在对应的属性对象（对于集合，可能会包含进来很多乱七八糟的值）

1. 扫描类中所有属性，对于有set方法并且非简单的属性，按类型从容器中选择最佳的属性值，存储到PropertyValues中。
2. 存储依赖关系
3. 对于类中有@Autowire的属性，获取器属性值，存储到PropertyValues中（类中：所有@Autowire注解的属性）

首先得有对应的BPP，有BPP才能解析注解的属性值

20210803211504

主要有下面两种BPP：

CommonAnnotationBeanPostProcessor：为注解赋值

AutowiredAnnotationBeanPostProcessor：为注解赋值

举例：AutowiredAnnotationBeanPostProcessor

先获取class中的所有AutoWire注解的属性，然后从容器中获取或者创建属性值。

1. 根据前面PropertyValues中的Value（标签对应的属性值、auto-wired获取的属性值、@Autowire注解获取的属性值），往bean对象设置属性值（可以是基本数据类型，也可以是引用类型）

对于不同类型的属性值，需要根据有不同的解析方法，通过转换为实际的对象。

* 根据属性值的类型，通过对应的解析方法获取属性的实际值（如果是引用对象，会调用getBean创建需要依赖的对象）
* 所有属性解析完成之后，通过set方法往bean中设置实际属性值

总结：

bean名：Property类private final String name字段

属性值：Property类private final Object value字段（有11种类型：RuntimeBeanReference、BeanDefinitionHolder、BeanDefinition、DependencyDescriptor、ManagedArray、ManagedList、ManagedSet、ManagedMap、ManagedProperties、TypedStringValue、NullBean）

实际值：PropertyValue类的private Object convertedValue字段

属性值：

1. 创建BeanDefinition的时候，会确定<property>的属性值
2. 处理配置文件中Auto-wired=”by-name/type”的时候，会确定类中非简单属性属性值
3. 处理类中@Autowire等注解的时候，会确定注解属性的属性值

实际值：

通过上述获取到的属性值，根据属性值的不同类型，解析出属性的实际值，然后通过set方法设置到bean对象中。