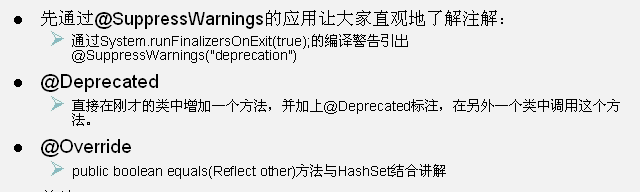
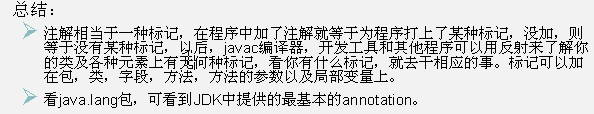
1. 了解注解及java提供的几个基本注解





1. JDK自带的注解
   1. @Override

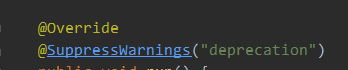
*Indicates that a method declaration is intended to override or implement a method declaration in a supertype*

* 1. @Deprecated

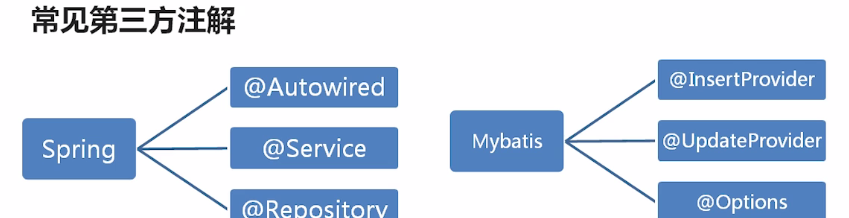
*programmers are discouraged from using, typically because it is dangerous, or because a better alternative exists. Compilers warn when a deprecated program element is used or overridden in non-deprecated code*

* 1. @SuppressWarnings

*Indicates that the named compiler warnings should be suppressed in the annotated element*

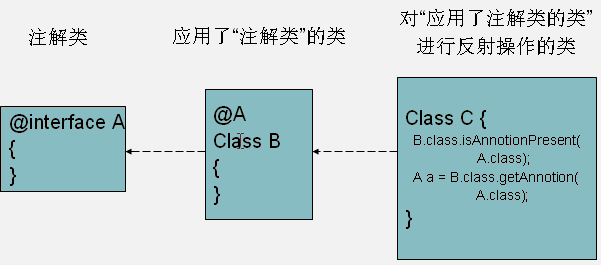


1. 第三方注解

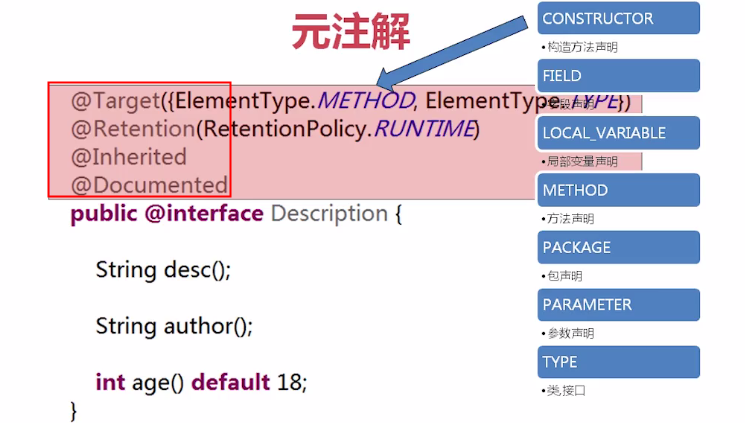


1. 注解的分类
   1. 按照运行机制分
      1. 源码注解(RetentionPolicy.SOURCE)：注解只在源码中存在，编译成.class文件就不存在了(jdk自带注解@Override、@ SuppressWarnings属于源码注解)：只在源码显示，编译时会丢弃
      2. 编译时注解(RetentionPolicy .CLASS)：注解在源码和.class文件中都存在：编译时会记录到class中，运行时忽略
      3. 运行时注解(RetentionPolicy.RUNTIME)：在运行阶段还起作用，甚至会影响运行逻辑的注解(@Deprecated、@Autowired等属于运行时注解)：运行时存在，可以通过反射读取
   2. 按照来源分
      1. 来自JDK的注解
      2. 来自第三方的注解
      3. 自定义的注解
2. 注解的应用结构图

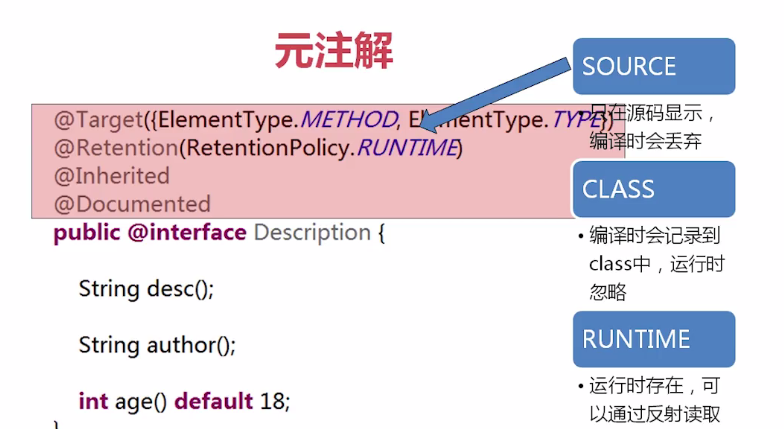
注解就相当于一个你的一个源程序中要调用的一个类，要在源程序中应用某个注解，得先准备好了这个注解类。就相当于你要调用某个类，得先开发好这个类。



1. 元注解：注解的注解
   1. @Target



* 1. @Retention

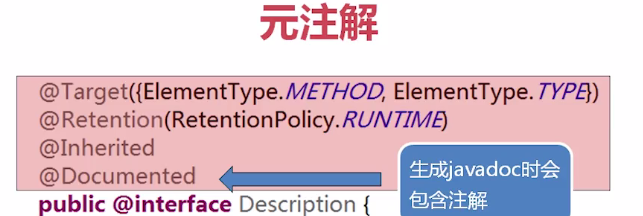


* 1. @Inherited



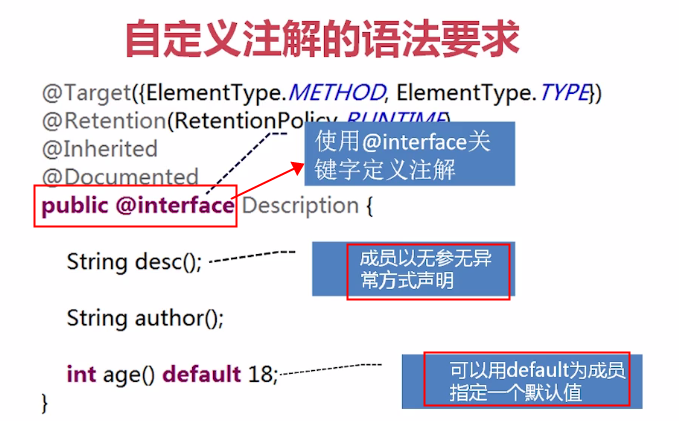
标记在父类中的注解可以在子类中获取

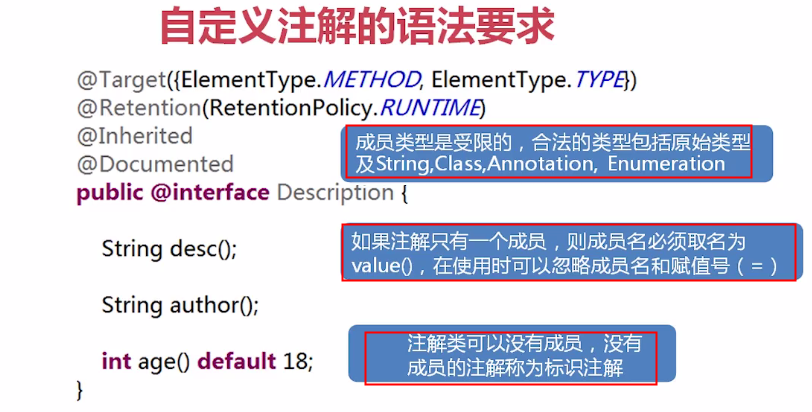
* 1. @Documented



1. 自定义注解及其应用

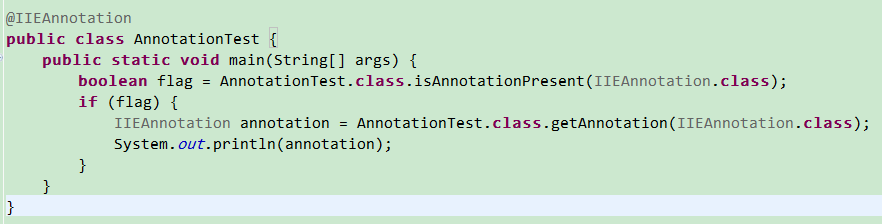
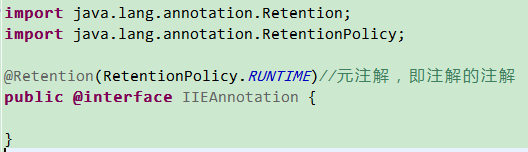






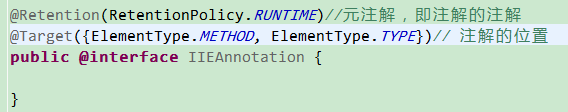
默认注解生命周期在CLASS阶段。

@Override、@SuppressWarning生命周期在SOURCE阶段，@Deprecated生命周期在RUNTIME阶段

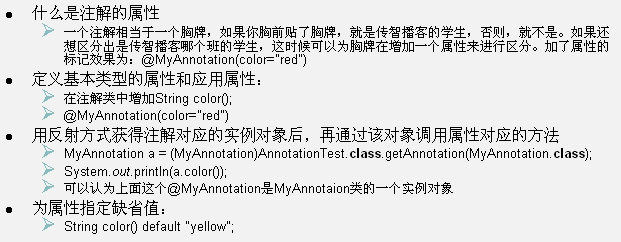


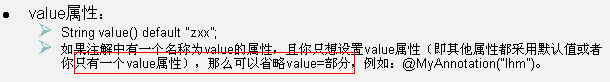
@Target注解

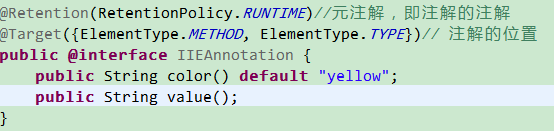




1. 为注解增加基本属性

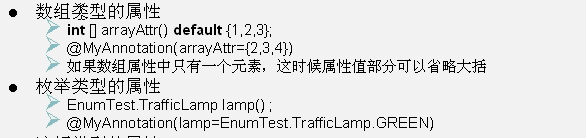








1. 为注解增加高级属性





1. 使用自定义注解

