目录

1.说说你对 Java 注解的理解（源码）

java 注解：

注解，也叫元数据。可以声明在包、类、字段、方法、局部变量、方法参数等前面，来对这些元素进行说明，注释等。

元注解：

java 提供了四个元注解，所谓元注解就是负责注解其他注解。

1）**@Target** ：规定注解所修饰的对象范围。

① ElementType.CONSTRUCTIR; 构造器声明

② ElementType.FIELD; 成员变量，对象，属性（包括 enum 实例）

③ ElementType.LOCAL_VARIABLE; 局部变量声明

④ ElementType.METHOD ; 方法声明

⑤ ElementType.PACKAGE; 包声明

⑥ ElementType.PARAMETER;参数声明

⑦ ElementType.TYPE; 类、接口（包括注解类型）或 enum 声明

2）**@Retention** ： 表示注解的生命周期

① RetentionPolicy.SOUREC: 在源文件中有效

② RetentionPolicy.CLASS; 在 class 文件中有效

③ RetentionPolicy.RUNTIME;在运行时有效

3) **@Inherited**：标记注解，主要说明了一种继承性，意思是子类可以继承父类中的该注解（注意：只有当被贴上@Inherited 标签的注解被用在类上的时候子类才能获得这个注解）。

4) **@Documented**： 用于描述其它类型的 annotation（注解）应该被作为被标注的程序成员的公共 API,因此可以被例如 javadoc 此类的工具文档化。（表明这个注释是由 javadoc 记录的，在默认情况下也有类似的记录工具。 如果一个类型声明被注释了文档化，它的注释成为公共 API 的一部分。）

2.说说你对依赖注入的理解（源码）

自己看总结
https://blog.csdn.net/bestone0213/article/details/47424255

DI—Dependency Injection，即“依赖注入”：组件之间依赖关系由容器在运行期决定（由容器动态的将某个依赖关系注入到组件之中）。

依赖注入的目的并非为软件系统带来更多功能，而是为了提升组件重用的频率，并为系统搭建一个灵活、可扩展的平台。通过依赖注

入机制，我们只需要通过简单的配置，而无需任何代码就可指定目标需要的资源，完成自身的业务逻辑，而不需要关心具体的资源来自何处，由谁实现。

理解 DI 的关键是：“谁依赖谁，为什么需要依赖，谁注入谁，注入了什么”，那我们来深入分析一下：

1) 谁依赖于谁：当然是应用程序依赖于 IoC 容器；

2) 为什么需要依赖：应用程序需要 IoC 容器来提供对象需要的外部资源；

3) 谁注入谁：很明显是 IoC 容器注入应用程序某个对象，应用程序依赖的对象；

4) 注入了什么：就是注入某个对象所需要的外部资源（包括对象、资源、常量数据）。

IoC 的一个重点是在系统运行中，动态的向某个对象提供它所需要的其他对象。这一点是通过 DI（Dependency Injection，依赖注入）来实现的。比如对象 A 需要操作数据库，以前我们总是要在 A 中自己编写代码来获得一个 Connection 对象，告诉依赖注入框架，A 中需要一个 Connection，至于这个 Connection 怎么构造，何时构造，A 不需要知道。

在系统运行时，依赖注入框架会在适当的时候制造一个 Connection，然后像打针一样，注射到 A 当中，这样就完成了对各个对象之间关系的控制。A 需要依赖 Connection 才能正常运行，而这

个 Connection 是由依赖注入框架注入到 A 中的，依赖注入的名字就这么来的。

可以看看 dagger2