IOC 和 DI 以及 AOP

作者：HuixiaZhang
原文链接：https://ld246.com/article/1539003482143
来源网站：链滴

1.IOC（Inverse of Control）：控制反转，也可以称为依赖倒置。

所谓依赖，从程序的角度看，就是比如A要调用B的方法，那么A就依赖于B，反正A要用到B，则A赖于B。所谓倒置，你必须理解如果不倒置，会怎么着，因为A必须要有B，才可以调用B，如果不倒，意思就是A主动获取B的实例：B b = new B()，这就是最简单的获取B实例的方法（当然还有各种计模式可以帮助你去获得B的实例，比如工厂、Locator等等），然后你就可以调用b对象了。所以，倒置，意味着A要主动获取B，才能使用B；到了这里，就应该明白了倒置的意思了。倒置就是A要调用的话，A并不需要主动获取B，而是由其它人自动将B送上门来。

形象的举例就是：

通常情况下，假如你有一天在家里口渴了，要喝水，那么你可以到你小区的小卖部去，告诉他们，需要一瓶水，然后小卖部给你一瓶水！这本来没有太大问题，关键是如果小卖部很远，那么你必须知：从你家如何到小卖部；小卖部里是否有你需要的水；你还要考虑是否开着车去；等等等等，也许有多的问题要考虑了。也就是说，为了一瓶水，你还可能需要依赖于车等等这些交通工具或别的工具，题是不是变得复杂了？那么如何解决这个问题呢？

解决这个问题的方法很简单：小卖部提供送货上门服务，凡是小卖部的会员，你只要告知小卖部你要什么，小卖部将主动把货物给你送上门来！这样一来，你只需要做两件事情，你就可以活得更加轻自在：

第一：向小卖部注册为会员。

第二：告诉小卖部你需要什么。

这和Spring的做法很类似！Spring就是小卖部，你就是A对象，水就是B对象

第一：在Spring中声明一个类：A

第二：告诉Spring，A需要B

假设A是UserAction类，而B是UserService类

```
<bean id="userService" class="org.leadfar.service.UserService"/><bean id="documetService" class="org.leadfar.service.DocumentService"/><bean id="orgService" class="org.eadfar.service.OrgService"/> <bean id="userAction" class="org.leadfar.web.UserAction"> property name="userService" ref="userService"/></bean>
```

在Spring这个商店（工厂）中，有很多对象/服务：userService,documentService,orgService也有很多会员：userAction等等，声明userAction需要userService即可，Spring将通过你给它提供通道主动把userService送上门来，因此UserAction的代码示例类似如下所示：

```
package org.leadfar.web;public class UserAction{ private UserService userService; public String login(){ userService.valifyUser(xxx); } public void setUserService(UserService userService){ this.userService = userService; }}
```

在这段代码里面，你无需自己创建UserService对象（Spring作为背后无形的手，把UserService象通过你定义的setUserService()方法把它主动送给了你，这就叫依赖注入！），当然咯，我们也可使用注解来注入。Spring依赖注入的实现技术是：动态代理

2.AOP：即面向切面编程

面向切面编程的目标就是分离关注点。什么是关注点呢？就是你要做的事，就是关注点。假如你是公子哥，没啥人生目标，天天就是衣来伸手，饭来张口，整天只知道玩一件事！那么，每天你一睁眼就光想着吃完饭就去玩（你必须要做的事），但是在玩之前，你还需要穿衣服、穿鞋子、叠好被子、

饭等等等等事情，这些事情就是你的关注点，但是你只想吃饭然后玩，那么怎么办呢？这些事情通通给别人去干。在你走到饭桌之前，有一个专门的仆人A帮你穿衣服，仆人B帮你穿鞋子，仆人C帮你叠被子，仆人C帮你做饭，然后你就开始吃饭、去玩（这就是你一天的正事），你干完你的正事之后，来，然后一系列仆人又开始帮你干这个干那个，然后一天就结束了！

AOP的好处就是你只需要干你的正事，其它事情别人帮你干。也许有一天，你想裸奔，不想穿衣服那么你把仆人A解雇就是了！也许有一天，出门之前你还想带点钱，那么你再雇一个仆人D专门帮你取钱的活！这就是AOP。每个人各司其职，灵活组合，达到一种可配置的、可插拔的程序结构。

从Spring的角度看，AOP最大的用途就在于提供了事务管理的能力。事务管理就是一个关注点，的正事就是去访问数据库，而你不想管事务（太烦），所以，Spring在你访问数据库之前，自动帮你启事务，当你访问数据库结束之后，自动帮你提交/回滚事务！

3.DI(Dependency Injection，依赖注入),

举个例子：我们一个登陆验证操作需要连接数据库查询比对数据，如果没有Spring，通过简单的Servlt实现的话，我们的步骤可能是：首先新建数据库连接，然后才能操作数据库，进行验证...有了Spring之后，这个建立连接的步骤就可以交由Spring来操作，至于什么时候建立，什么时候关闭，我不用管我只要在我要用这个连接的时候，由Spring注入给我就行了（类似于打针一样），Spring会在适当的候新建这个连接，然后注入到需要这个连接的类中。这个类需要依赖这个连接才能正常运作，而这个接不是自己新建的而是由Spring容器帮忙新建然后注入到类中的，这就是依赖注入名字的由来。