## Spring产生的历史背景

2007年基于java的一套开源框架Spring横空出世，它的出现大大的加快了j2ee的开发速度。程序员们只需要关心业务怎么完成，不在需要考虑之前的代码准备工作。

Spring框架除了帮我们管理对象及其依赖关系，还提供了很多实用的功能，像通用日志记录、性能统计、安全控制、异常处理等面向切面的能力，还能帮我们管理最头疼的数据库事务，本身提供了一套简单的JDBC访问实现，提供与第三方数据访问框架集成（如Hibernate\JPA\MyBatis），与各种Java EE 技术整合(Java Mail、任务调度Quartz等等)，提供一套自己的Web层框架Spring MVC,而且还能非常简单的与第三方Web框架集成，从这里我们可以认为Spring是一个超级大的粘合平台，俗称万能胶，除了自己提供功能外，还提供粘合其他技术和框架的能力，从而使我们可以更自由的选择到底使用什么技术进行开发。而且不管是Java SE（C/S架构）应用程序还是Java EE(B/S架构)应用程序都可以使用这个平台进行开发。如今的Spring已经不再是一个框架，早已经成为一种生态。SpringBoot的便捷式开发实现了约定优于配置，SpringCloud微服务的生态，提供了非常方便的解决方案（网关、配置中心，注册中心，服务协调与治理等等）

## Spring的特点

### 容器

Spring作为一个容器，可以管理对象的生命周期、对象与对象之间的依赖关系。可以通过配置文件来定义对象，以及设置其他对象的依赖关系。

理解：从代码上看，容器其实就是实现ApplicationContext接口的上下文对象，它起到对bean的初始化和管理。在代码上我们可以通过new ClassPathXmlApplicationContext对象或者FileSystemXmlApplication对象**来初始化容器**。在SpringMVC中我们使用web.xml配置文件来初始化容器。想要使用自己的配置文件，就需要在main方法里写入

ApplicationComtext application=

new ApplicationContext(new String[]{“server.xml”})即可

### IOC

* 1. IOC（控制反转）：程序中各个组件之间的关系，不由程序代码直接操控，而由容器控制。控制权由应用代码中转到了外部容器，即所谓的反转。
  2. DI（依赖注入）：各个组件之间的依赖关系不是在程序中写死，而是写在外部的配置文件中，然后运行时通过容器动态注入。

### AOP(切面式编程)

解释：在开发中我们难免会遇到各个业务中都要重复编写的代码，这段代码存在于业务代码的前面、后面、异常时等，这样的代码不适合用工具类方法提取出，需要使用动态代理来实现这些重复的功能代码。AOP就是为了解决这个问题而存在的。

一些概念：

1. 切面Aspect： @Aspect注解一般是用在类上，就可以理解为我们要找的大方向，比如这个注解使用在LoginService上
2. 切点PointCut：@PointCut使用在方法上，也就是我们确定大方向后具体某一个点，比如说LoginService类中的login方法
3. 织入Waving：可以理解为将公共代码写在业务代码的前面后面或者异常时等。常见的有@Before @AfterReturning @Around

注：部分内容摘抄自<https://www.cnblogs.com/create-my-future/p/11233940.html>