# 关于框架学习的建议

关于这些框架，工具的学习，我这里有一个建议：**什么都别学，从基础开始，把根基搞扎实了。然后自己动手做工程，遇到问题了，再去学习会更好。**我觉得一个好的老师，不是手把手地把知识教给你，而是能够指出问题的所在。我举一个例子。我之前指导过一个师弟学习Java，我让他自己用socke t写一个web server ，他使用了一个线程一个连接的原始的线程方案，并且使用本机测试，开了十来个客户端，觉得还不错。我就告诉他，你这样的测试是不对的，你要去找一个client端的benchm ark，或者自己写个工具，创建成百上千个连接，然后再看看你的web server的表现如何。后来，他发现，当创建几百个连接以后，服务端几乎就跑不动了。然后，我就提醒他，可以考虑一下，业界有哪些解决方案。这时候，他找到了nio，netty， 线程池等等方案。然后自己去重构，重构过程中，发现逻辑和页面合在一起，很难处理，又引入了spring mvc，为了操作数据库，又引入了hibernate等等。

回顾他学习的过程可以发现，**对于工具和框架，被动地学习会比主动地学习效果好。到了不得不学的时候再去学习，有具体的场景，学起来必然事半功倍。**

# 什么是Spring框架

spring是J2EE应用**程序框架，是轻量级**的**IoC和AOP**（面向切面编程）的容器框架，主要是针对javaBean的生命周期进行管理的**轻量级容器，可以单独使用**，也可以和Struts框架，ibatis框架等组合使用。

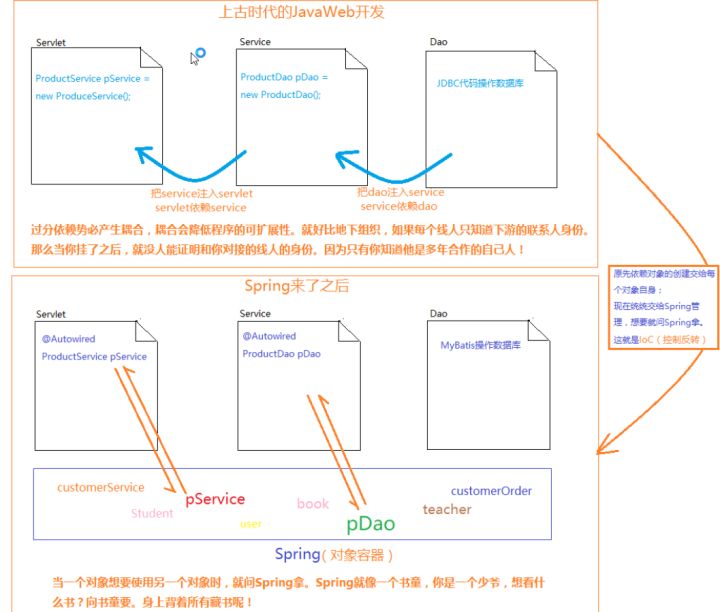
一句话，**Spring**是一个**开发应用框架**，什么样的框架呢，有这么几个标签：**轻量级、非侵入式、一站式、模块化**，其目的是用于**简化企业级应用程序开发**。我们知道应用程序是由一组相互协作的对象组成，而在传统应用程序开发中，一个完整的应用是由一组相互协作的对象组成。所以开发一个应用除了要开发业务逻辑之外，最多的是关注如何使这些对象协作来完成所需功能，而且要低耦合、高内聚。业务逻辑开发是不可避免的，那如果有个框架出来帮我们来创建对象及管理这些对象之间的依赖关系。当然，我们也可以通过自我能力，利用前人总结出来的设计模式来进行我们的架构设计，比如常见的「工厂模式」、「代理模式」等，总共有23种模式，利用别人总结的轮子来进行我们自己轮子建造，但如何这些轮子建造专门交给一个叫做Spring的人去帮你做，你只需要专心你的业务逻辑，是不是省心很多，同时这个叫Spring干的又专业又稳定，何乐而不为呢。

从这里我们可以认为**Spring**是一个超级粘合平台，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。

作者：黄孙挺  
链接：https://www.zhihu.com/question/21142149/answer/310738460  
来源：知乎  
著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

另外，Spring最核心的概念就两个：**AOP**（切面编程）和**DI**（依赖注入）。DI又依赖IoC。

这里不打算详细说，就简单画个图，只介绍DI和IoC。



简而言之，原先创建对象以及注入对象都是采取“人盯人”战术，一个咬着一个。有了Spring以后，通过IoC，所有的对象都可以从Spring容器中得到。每个对象由Spring注入到它应该去的地方。这种由原先对象管理切换到Spring管理的方式，就是所谓的IoC（控制反转），因为创建、管理对象的角色反过来了，有每个对象自主管理变为Spring统一管理。

而且，只有通过IoC先由Spring创建对象后，才能进行下一步对象注入（DI），所以说DI依赖IoC。

上面有说错的地方，请斧正。

## ioc的意义是啥？

面试的时候有人说是减少代码，这样的人是完全不懂。最起码答复到解耦，更加完整的是和AOP的关系，没有ioc就没有AOP，IOC相对于掐住了咽喉，对象是他创建的，他想干嘛就干嘛，才能实现AOP，才能增强特性。

# 为什么要用spring

在不使用spring框架之前，我们的service层中要使用dao层的对象，不得不在service层中new一个对象。如下：

**[java]** [view plain](https://blog.csdn.net/lishuangzhe7047/article/details/20740209) [copy](https://blog.csdn.net/lishuangzhe7047/article/details/20740209)

1. //dao层对象
2. **public** **class** UserDao{
3. publicvoid insert(User user){}
4. }
6. //service层对象
7. **public** classUserService{
8. publicvoid insert(User user){
9. UserDaouserdao = **new** UserDao();
10. userdao.insert(user);
11. }
12. }

存在的问题：层与层之间的依赖。

使用框架后：

**[java]** [view plain](https://blog.csdn.net/lishuangzhe7047/article/details/20740209) [copy](https://blog.csdn.net/lishuangzhe7047/article/details/20740209)

1. //dao层对象
2. **public** **class** UserDao{
3. publicvoid insert(User user){}
4. }
6. //service层对象
7. **public** classUserService{
8. privateUserDao userdao;
10. publicUserDao getUserdao() {
11. returnuserdao;
12. }
13. publicvoid setUserdao(UserDao userdao) {
14. **this**.userdao= userdao;
15. }
17. publicvoid insert(User user){
18. userdao.insert(user);
19. }
21. }

service层要用dao层对象需要配置到xml配置文件中，至于对象是怎么创建的，关系是怎么组合的都交给了spring框架去实现。

# 怎么学习

# 框架优点

轻量级的容器框架没有侵入性

使用IoC容器更加容易组合对象直接间关系，面向接口编程，降低耦合

Aop可以更加容易的进行功能扩展，遵循ocp开发原则

创建对象默认是单例的，不需要再使用单例模式进行处理

# 缺点：

业务功能依赖spring特有的功能，依赖与spring环境。